Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: Soluciones informáticas
Recibido: 10/05/2020 | Aceptado: 18/07/2020 | Publicado: 01/08/2020

Implementación del algoritmo meta-heurístico Gray Wolf Optimization para la optimización de funciones objetivo estándar

Implementation of the meta-heuristic algorithm Gray Wolf Optimization for the optimization of standard objective functions

Randy Alonso Benítez [1*], Jairo Rojas Delgado [2]

[1] Universidad de las Ciencias Informáticas (UCI). randy.alonso@nauta.cu
[2] Centro de Estudios de Matemática Computacional (CEMC). Universidad de las Ciencias Informáticas (UCI). jrdelgado@uci.cu

[*] Autor para correspondencia: randy.alonso@nauta.cu

Resumen

En la actualidad surgen implementaciones de algoritmos metaheurísticos para dar soluciones a problemas reales en la vida cotidiana, debido a su capacidad de identificar regiones prometedoras e intensificar la búsqueda en las mismas. En este trabajo se realizó un estudio de los algoritmos metaheurísticos recientes en la literatura y específicamente del algoritmo Gray Wolf Optimization. Se implementó el algoritmo meta-heurístico Gray Wolf Optimization para la optimización de funciones objetivo estándar, en el lenguaje C++, se diseñaron y se aplicaron las pruebas de eficiencia y precisión arrojando como resultado que, el algoritmo Gray Wolf Optimization obtuvo mayor precisión al ser comparado con otros algoritmos metaheurísticos a pesar de no poseer hıper-parámetros ajustables a cada problema de optimización. Adicionalmente este algoritmo es menos eficiente que los algoritmos estudiados, no obstante, la diferencia observada en tiempo de ejecución se encuentra en el orden de los milisegundos. Se obtuvo como resultado un producto informático que favorecerá el mejoramiento de la optimización de funciones objetivo estándar para el desarrollo de futuras aplicaciones.

Palabras clave: optimización, algoritmo, funciones, heurística.

Abstract

Nowadays, implementations of metaheuristic algorithms are very often emerging to provide solutions to real problems in everyday life, due to their ability to identify promising regions and intensify the search in them. A thorough study of recent metaheuristic algorithms in the literature was conducted. An in-depth study of the Gray Wolf Optimization algorithm was carried out, the Gray Wolf Optimization meta-heuristic algorithm was implemented for the optimization of standard objective functions, it was implemented in the C ++ language, efficiency and precision

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

tests were designed and applied, yielding as a result that, the GWO algorithm obtained greater precision when compared with the PSO and FA algorithms despite not having hyper-adjustable parameters to each optimization problem and additionally this algorithm is less efficient than the PSO and FA algorithms, however, the difference observed at run time it is in the order of milliseconds. As a result, a computer product was obtained that will favor the improvement of the optimization of standard objective functions for the development of future applications.

Keywords: optimization, algorithm, functions, heuristics

Introducción

Un algoritmo meta-heurístico de optimización permite explorar un espacio de búsqueda sin garantías de encontrar una solución o que dicha solución sea óptima. El principal criterio que define el éxito de estos algoritmos es la capacidad de equilibrar la exploración en el espacio de búsqueda con la explotación de la información local, es decir, la capacidad de identificar regiones prometedoras e intensificar la búsqueda en las mismas. Debido a la naturaleza estocástica de estos algoritmos, para su evaluación es necesario la experimentación y su estudio en la práctica. Entiéndase por evaluación de un algoritmo meta-heurístico un análisis de la velocidad de convergencia y precisión de los mismos.

El algoritmo meta-heurístico Gray Wolf Optimization (GWO) (Mirjalili, 2014) fue introducido recientemente inspirado en el comportamiento de los lobos grises en la naturaleza. De la misma forma, su empleo en la optimización de funciones objetivo estándar ha sido estudiado en trabajos recientes.

Adicionalmente, contrastar las diferencias estadísticas de la eficiencia y precisión del algoritmo GWO en la optimización de funciones objetivo estándar respecto a otros algoritmos meta-heurísticos resulte de sumo interés práctico para futuras aplicaciones. Estos algoritmos son sensibles a cambios en la implementación, una pequeña variación en un detalle, digamos la distribución de probabilidad a partir de la cual se generan los números aleatorios, causa comportamientos diferentes del mismo. La selección de los híper-parámetros es otro de los puntos sensibles de estos algoritmos, los mismos influyen en la eficiencia y precisión de estos algoritmos y por lo tanto es una variable ajena a controlar. Los dos factores anteriores hacen que evaluar estos algoritmos, incluyendo GWO sea complejo. Dos implementaciones del mismo algoritmo pueden arrojar resultados diferentes. Esto es se observa con mayor influencia en algoritmos recientes, menos establecidos, donde no existe una cantidad significativa de implementaciones y aclaraciones en la literatura sobre sus detalles de implementación. Un escenario similar puede encontrarse al comparar dos algoritmos meta-heurísticos usando implementaciones de diferentes orígenes, donde los detalles de implementación tienen la capacidad de sesgar los resultados.

En el Centro de Estudios de Matemática Computacional perteneciente a la Universidad de las Ciencias Informáticas, se desarrolló una biblioteca de clases (dnn_opt). Esta biblioteca está implementada en C++11 y posee implementaciones eficientes para optimizar funciones objetivo estándar a partir de algoritmos meta-heurísticos de optimización, sin embargo, no cuenta con implementaciones para este algoritmo.

Materiales y métodos

En este epígrafe se describe el concepto del término metaheurística y los principales autores que lo definieron y su clasificación, seguidamente se describe detalladamente qué es el algoritmo GWO, su inspiración y modelación matemática y se describe cómo este algoritmo hace los procesos de exploración y explotación del espacio de búsqueda.

Algoritmos Meta-heurísticos

El término metaheurística o meta-heurística fue acuñado por F. Glover en el año 1986 (Sorensen, 2017). Con este término, pretendía definir un "procedimiento maestro de alto nivel que guía y modifica otras heurísticas para explorar soluciones más allá de la simple optimalidad local". Actualmente, existe una cantidad muy importante de trabajos científicos publicados que abordan problemas de optimización a través de las meta-heurísticas, investigaciones sobre nuevas meta-heurísticas o extensiones de las meta-heurísticas ya conocidas (Zheng, 2015)(Garcıa, 2016)(Mesejo, 2015)(Zheng, 2015).

A partir de la definición original de F. Glover, en la literatura se pueden encontrar otras definiciones alternativas de meta-heurísticas o heurísticas modernas, entre las que se pueden destacar las siguientes:

- Para Kelly, J.P.(1997).: "Las meta-heurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Las meta-heurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los procedimientos estadísticos" (Osman, 1997)
- Para Vob, S. (2001): "Una meta-heurística es un proceso iterativo maestro que guía y modifica las operaciones de una heurística subordinada para producir eficientemente soluciones de alta calidad. Las metaheurísticas pueden manipular una única solución completa (o incompleta) o una colección de soluciones en cada iteración. La heurística subordinada puede ser un procedimiento de alto (o bajo) nivel, una búsqueda local, o un método constructivo" (S. Voß , 2000). La idea básica general es siempre la misma: enriquecer a los algoritmos heurísticos de forma que éstos no se queden atrapados en óptimos locales.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

La evolución de las metaheurísticas durante los últimos 25 años ha tenido un comportamiento prácticamente exponencial. En el tiempo que transcurre desde las primeras reticencias hasta la actualidad, se han encontrado soluciones de muy alta calidad a problemas que hace tiempo parecían inabordables.

De modo general, se puede decir que las metaheurísticas combinan ideas que provienen de cuatro campos de investigación bien distintos.

- Las técnicas de diseño de algoritmos (resuelven una colección de problemas).
- Algoritmos específicos (dependientes del problema que se quiere resolver).
- Fuente de inspiración (del mundo real).
- Métodos estadísticos.

Una primera conclusión que se puede extraer de las definiciones dadas es que, en muchos casos, son reglas de sentido común que permiten hacer una búsqueda "inteligente". Debido a esta característica, para bastantes metaheurísticas no existe un marco teórico que las sustente, sino que es a través de los buenos resultados experimentales donde encuentran su justificación.

Las metaheurísticas se pueden clasificar de la siguiente manera:

Atendiendo a la Inspiración:

- Natural: algoritmos que se basan en un símil real, ya sea biológico, social, cultural, etc.
- Sin inspiración: algoritmos que se obtienen directamente de sus propiedades matemáticas.

Atendiendo al número de soluciones:

- Poblacionales: buscan el óptimo de un problema a través de un conjunto de soluciones.
- Trayectoriales: trabajan exclusivamente con una solución que mejoran iterativamente.

Atendiendo a la función objetivo:

- Estáticas: no hacen ninguna modificación sobre la función objetivo del problema.
- Dinámicas: modifican la función objetivo durante la búsqueda.

Atendiendo a la vecindad:

- Una vecindad: durante la búsqueda utilizan exclusivamente una estructura de vecindad.
- Varias vecindades: durante la búsqueda modifican la estructura de la vecindad.

Atendiendo al uso de memoria:

- Sin memoria: se basan exclusivamente en el estado anterior.
- Con memoria: utilizan una estructura de memoria para recordar la historia pasada.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

Adicionalmente, los resultados teóricos derivados de la adaptación de los teoremas no-free-lunch para las metaheurísticas sugieren que, ningún algoritmo meta-heurístico es estadísticamente superior a otro al considerar el conjunto de todos los problemas de optimización, de tal forma que si un algoritmo A es más eficiente que un algoritmo B en un conjunto de problemas, debe existir otro conjunto de problemas de igual tamaño para los que el algoritmo B sea más eficiente que el A.

Por último, las metaheurísticas más optimizadas son demasiado dependientes del problema o al menos necesitan tener un elevado conocimiento heurístico del problema. Esto hace que, en general, se pierda la genericidad original con la que fueron concebidas.

A pesar de estos aparentes problemas, la realidad es que el comportamiento experimental de la mayoría de las metaheurísticas es extraordinario, convirtiéndose para muchos problemas difíciles de resolver en la única alternativa factible para encontrar una solución de calidad en un tiempo razonable. En general, las metaheurísticas se comportan como métodos muy robustos y eficientes que se pueden aplicar con relativa facilidad a una colección amplia de problemas. Además, la demostración del teorema NFL se basa en que el algoritmo de búsqueda no visita dos veces la misma solución y en que no se introduce conocimiento heurístico en el diseño del método metaheurístico. Estas hipótesis habitualmente no son ciertas.

Grey Wolf Optimizer

El algoritmo Grey Wolf Optimizer es un algoritmo meta-heurístico de optimización, basado en inteligencia colectiva y desarrollado por Seyed Mohammad Mirjalili en el año 2014 (Mirjalili, 2014), está inspirado en el comportamiento social de las manadas de lobos grises (Canis Lupus) y su organización para la actividad de caza de presas. En el presente epígrafe se modela matemáticamente el algoritmo GWO y se describe detalladamente su inspiración, también se describen los procesos de exploración y explotación así como su pseudocódigo.

Inspiración del algoritmo Gray Wolf Optimizer

Los lobos grises viven y cazan en manadas de alrededor de seis a diez miembros como promedio, recorren grandes distancias, alrededor de doce millas en un solo día. El territorio de una manada de lobos puede cubrir 20-80 millas cuadradas. Estos animales sociales cooperan en la caza de sus presas preferidas, animales grandes como ciervos, alces y alces. Cuando tienen éxito, los lobos no comen con moderación. Un solo animal puede consumir 20 libras de carne en una sesión. En cada manada de lobos grises, hay una jerarquía social común que dicta poder y dominación.

La manada se conforma de un líder llamado alfa y los miembros beta, delta y omega. En esta estructura social, en un primer nivel, el alfa es el responsable de la toma de decisiones sobre la caza, lugar donde dormir, hora de despertar, y

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

el resto de los miembros debe seguir sus órdenes. El miembro alfa de una ma-nada no necesariamente debe ser el más fuerte, pero sí el mejor en términos de conducción del grupo y capacidad para tomar decisiones.

El segundo nivel en la jerarquía del grupo son los miembros beta (subordinados de los alfa), que deben colaborar con la toma de decisiones y el cumplimiento de las órdenes de los alfa. Estos miembros son los candidatos a convertirse en alfa cuando alguno de estos fallezca o envejezca.

En un tercer nivel se encuentran los lobos delta. Los lobos delta tienen que someterse a alfas y betas. Este nivel está compuesto por exploradores, centinelas, ancianos, cazadores y cuidadores. Los exploradores son responsables de vigilar los límites del territorio y de advertir a la manada en caso de algún peligro. Los centinelas protegen y garantizan la seguridad del paquete. Los ancianos son lobos experimentados que solían ser alfa o beta. Los cazadores ayudan a los alfas y betas cuando cazan presas y proporcionan comida para la manada. Finalmente, los cuidadores son responsables de cuidar a los lobos débiles, enfermos y heridos en la manada.

Existe otra clasificación que se encuentra más abajo en la jerarquía de las manadas de lobos grises y se denomina omega, estos deben someterse al alfa, beta y delta. En muchos casos suele verse en este tipo a miembros que ofician de niñeros de las crías dentro de la manada.

Esta jerarquía es el pilar fundamental de inspiración del algoritmo GWO, en la cual, las tres mejores soluciones en una población están representadas por los lobos alpha, beta y delta. Otro factor de suma importancia inspiración de este algoritmo es el enfoque de caza de los lobos grises. Al cazar una presa, los lobos grises siguen una serie de pasos eficientes:

- Rastreando, persiguiendo y acercándose a la presa.
- Perseguir, rodear y acosar a la presa hasta que deje de moverse.
- Ataque hacia la presa.

Para diseñar el algoritmo GWO es necesario modelar matemáticamente la jerarquía social de los lobos grises y su estilo de caza.

Modelo matemático del algoritmo GWO

Jerarquía social

Para modelar matemáticamente la jerarquía social de los lobos, se denota como el conjunto $X = \{\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_p\}$ a la población de lobos donde p es la cantidad de lobos de la población y $\vec{X}_i$ un individuo de la misma donde $\vec{X}_i \in R^n$, siendo la cantidad de dimensiones del problema de optimización. Teniendo en cuenta el valor de la función objetivo $f: R^n \rightarrow R$ se considera $\vec{X}_\alpha$ al mejor individuo de la población. En consecuencia, las segunda y tercera mejor solución

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

se nombran $\vec{X}_\beta$ y $\vec{X}_\delta$ respectivamente. El resto de las soluciones candidatas son asumidas como omega (ω). La determinación de alpha, beta y delta se determina de acuerdo al valor de la función objetivo, es decir: f ($\vec{X}_\alpha$)< f ($\vec{X}_\beta$)< f ($\vec{X}_\delta$) para problemas de minimización.

Caza

La jerarquía social juega un papel clave en la caza y la supervivencia de un grupo. Para simular la jerarquía social, las tres mejores soluciones se consideran alfa ($\vec{X}_\alpha$), beta($\vec{X}_\beta$) y delta($\vec{X}_\delta$). Aunque en la naturaleza puede haber más de un lobo en cada categoría, se considera que solo hay una solución para cada clase en GWO en aras de la simplicidad.
La posición del óptimo global se aproxima a partir de la posición del alpha, beta y delta que en su conjunto son las tres mejores soluciones de la población. De esta forma los lobos actualizan sus posiciones de acuerdo a la ecuación.

$$\vec{X}_i^{(t+1)} = \frac{\vec{X}_\alpha + \vec{X}_\beta + \vec{X}_\delta}{3} \quad (1)$$

donde $\vec{X}_i^{(t+1)}$ es la siguiente posición de un lobo, $\vec{X}_i$ se definen de forma tal que encierren en una hiperesfera el óptimo global.
Rodeando la presa
Para modelar matemáticamente el proceso de rodear a la presa, GWO considera la posición de cada lobo en un espacio n-dimensional y basado en la posición aproximada de la presa, es decir α, ($\vec{X}_\beta$) y ($\vec{X}_\delta$) de forma que:

$$\vec{X}^i = \vec{X}_j - \vec{A}_j \odot (\vec{D}_j) \quad (2)$$

Donde $\vec{X}_i$ es la siguiente ubicación del lobo, $\vec{X}_j$ es la ubicación aproximada de la presa y donde representa el producto de Hadamart entre dos vectores. Siendo $\vec{A}_i$ y $\vec{D}_i$ dos vectores que se definen de la siguiente manera:

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

$$\vec{D}_j = | \vec{C}_j \cdot \vec{X}_j - \vec{X}_i^t | \quad (3)$$

$$\vec{A} = 2a^t \vec{r_1} - a \cdot e \quad (4)$$

Donde || se refiere al valor absoluto, $\vec{X}_i$ es la ubicación actual del lobo, a es un número real donde sus valores disminuyen linealmente de 2 a 0 durante el transcurso de la ejecución, e es el vector de unos de dimensión n. $\vec{C}_j$ es un vector de coeficientes que se calcula de la siguiente forma:

$$\vec{C}_j = 2 \cdot \vec{r_2} \quad (5)$$

donde $\vec{r}_1$ y $\vec{r}_2$ son vectores generados aleatoriamente con una distribución uniforme en el intervalo $\vec{r}_i \in [0, 1]$. La ecuación para actualizar el parámetro a es la siguiente:

$$a = a \cdot 0.99 \quad (6)$$

Los vectores aleatorios $\vec{r}_1$ y $\vec{r}_2$ permiten a los lobos alcanzar cualquier posición entre los puntos ilustrados en la **Figura 1**. Por lo tanto, un lobo gris puede actualizar su posición dentro del espacio alrededor de la presa en cualquier ubicación aleatoria mediante el uso de las ecuaciones 2 y 3.

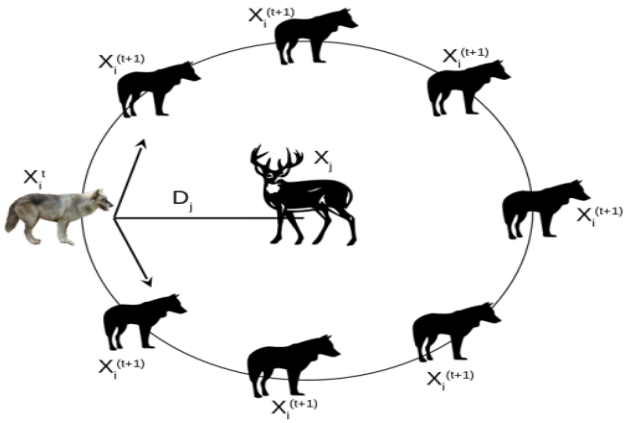

Figura 1. Cómo las ecuaciones matemáticas permiten actualizar la posición alrededor de un punto de pivote.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

La **ecuación 2** modela matemáticamente la actualización de posición de un lobo gris $\vec{X}_i$ alrededor de una presa $\vec{X}_j$ .Dependiendo de la distancia entre el lobo y la presa (D_j), un lobo puede ser reubicado en un círculo (en un espacio 2D), una esfera (en un espacio 3D) o una hiperesfera (en un espacio N-dimensional) alrededor de la presa

Explotación y exploración en el algoritmo GWO

La exploración y la explotación son dos procesos en conflicto que un algoritmo puede mostrar al optimizar un problema determinado. En el proceso de exploración, el algoritmo intenta descubrir nuevas partes del espacio de búsqueda de problemas mediante la aplicación de cambios repentinos en las soluciones, ya que el objetivo principal es descubrir las áreas prometedoras del espacio de búsqueda y evitar que las soluciones se detengan en un óptimo local.

En la explotación, el objetivo principal es mejorar las soluciones estimadas logradas en el proceso de exploración descubriendo el vecindario de cada solución. Por lo tanto, se deben hacer cambios graduales en las soluciones para converger hacia el óptimo global. El principal desafío aquí es que la exploración y la explotación están en conflicto. Por lo tanto, un algoritmo debería poder abordar y equilibrar estos comportamientos conflictivos durante la optimización para encontrar una estimación precisa del óptimo global para un problema dado.

El principal parámetro de control de GWO para promover la exploración es la variable $\vec{C}_j$. Este parámetro siempre devuelve un valor aleatorio en el intervalo de [0, 2] en todo momento para enfatizar la exploración no solo durante las iteraciones iniciales sino también las iteraciones finales. Esto ayuda a GWO a mostrar un comportamiento más aleatorio a lo largo de la optimización, favoreciendo la exploración y la evitación de los óptimos locales. El vector $\vec{C}_j$ también puede considerarse como el efecto de los obstáculos para acercarse a la presa en la naturaleza. En términos generales, los obstáculos en la naturaleza aparecen en los caminos de caza de los lobos y, de hecho, evitan que se acerquen rápida y convenientemente a sus presas. Esto es exactamente lo que hace el vector $\vec{C}_j$. Dependiendo de la posición de un lobo, puede darle un peso aleatorio a la presa y hacer que sea más difícil y más lejos de alcanzar a los lobos, o viceversa.

Dado que estos parámetros proporcionan valores aleatorios independientemente del número de iteración, beneficia la elución de un óptimo local, especialmente en las iteraciones finales.

Otro parámetro de control que favorece la exploración es $\vec{A}_j$ El valor de este parámetro se define por a, que disminuye linealmente de 2 a 0. Debido a los componentes aleatorios en este parámetro, el rango cambia en el intervalo de [-2, 2] para el parámetro $\vec{A}_j$. La exploración se promueve cuando $\vec{A}_j$< 1 o $\vec{A}_j$>-1, mientras que hay énfasis en la explotación cuando [-1 < $\vec{A}_j$< 1].

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

Se requiere un buen equilibrio entre exploración y explotación para encontrar una aproximación precisa del óptimo global utilizando algoritmos estocásticos. Este equilibrio se realiza en GWO con el comportamiento decreciente del parámetro a en la ecuación para el parámetro $\vec{A}_j$.

Pseudocódigo del algoritmo GWO

En este epígrafe se detalla en pseudocódigo los pasos para buscar el óptimo global en una población de lobos, nombrado como GWO.

Algoritmo 2.1: Gray Wolf Optimizer

```
1: Inicializar la población de lobos X_i siendo i = 1, ..., n aleatoriamente
2: Inicializar los valores de a, A y C
3: Calcular el fitness de cada agente de búsqueda
4: X_α = mejor agente
5: X_β = segundo mejor agente
6: X_δ = tercer mejor agente
7: while t < máximo número de iteraciones do
8:     for i ← 1 : n do
9:         Actualizar la posición de cada agente
10:    end for
11:    Actualizar los valores de a, A y C
12:    Calcular el fitness de todos los agentes de búsqueda
13:    Actualizar X_α, X_β, X_δ
14:    t = t+1
15: end while
16: return X_α
```

Figura 2. Pseudocódigo del algoritmo Gray Wolf Optimizer

En el paso 1 se inicializa la población de lobos a optimizar. En el paso 2 se inicializan los parámetros de control de los procesos de exploración y explotación del algoritmo, en los que se intenta descubrir áreas prometedoras en el espacio de búsqueda para converger hacia el óptimo global y evitar que las soluciones se detengan en un óptimo local.

En el paso 3 se calcula el valor de la función objetivo para cada agente en la población y posteriormente en los pasos 4, 5 y 6, se extraen cuáles son los tres mejores agentes respectivamente hasta el momento.

En el paso 8 se recorre la población a optimizar. En el paso 9 se actualiza la posición de cada agente de la población usando la ecuación 6.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

En el paso 11 se actualizan los valores de los parámetros a, $\vec{A}_j$ y $\vec{C}_j$ los cuales favorecen los procesos de exploración y explotación. Esto ayuda al algoritmo a presentar un comportamiento más aleatorio a lo largo de la optimización. En el paso 12 se recalcula el valor de la función objetivo para cada agente, para luego en el paso 13 actualizar los tres mejores agentes $\vec{X}_\alpha$, $\vec{X}_\beta$ y $\vec{X}_\delta$.

En el paso 14 actualiza en valor de la variable t la cual controla el transcurso de las iteraciones y posteriormente en el paso 16 se retorna el valor de mejor agente $\vec{X}_\alpha$.

Resultados y discusión

Diseño de las pruebas

Para la realización de las pruebas en el presente trabajo, se consideran variables independientes las funciones objetivos y el algoritmo de optimización. Ambas variables independientes son nominales donde cada nivel de la variable representa una función objetivo o algoritmo de optimización según corresponda. Adicionalmente, se identifican como variables dependientes la eficiencia y la precisión del proceso de optimización. Las variables dependientes son de naturaleza continua con nivel de medición de razón. Finalmente, se identifica como variable ajena el método de configuración de los híper-parámetros de los algoritmos meta-heurísticos, siendo esta una variable nominal.

Variables independientes

Funciones objetivos

En la literatura, existen varias funciones objetivos estándar para probar algoritmos de optimización. En este trabajo, se seleccionaron un subconjunto de estas sobre la base de cinco propiedades relacionadas con estas: continua / discontinua, diferenciable / no-diferenciable, separable/ no separable y unimodal / multimodal. (Liang, 2005)(Momin, 2013.)

1. De Jung:

$$f_1(w) = \sum_{i=0}^{n} w_i^2 \qquad (7)$$

posee un mínimo global en $f1(0, ..., 0) = 0$. De jung es continua, diferenciable, separable, separable y multimodal.

2. Ackley:

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

$$f_2(w) = -20e^{0.02\sqrt{n^{-1}\sum_{i=0}^{n} w_i^2}} - e^{n^{-1}\sum_{i=0}^{n}\cos(2\pi w_i)} + 20 + e \quad (8)$$

posee un mínimo global en $f2(0, ..., 0) = 0$. Ackley es continua, diferenciable, no-separable, separable y multimodal.

3. Grienwangk:

$$f_3(w) = \sum_{i=0}^{n} w_i^2/4000 - \prod_{i=0}^{n}\cos(w_i/\sqrt{i}) + 1 \quad (9)$$

posee un mínimo global en $f3(0, ..., 0) = 0$. Griendwangk es continua, diferenciable, no-separable, separable y multimodal.

4. Rastrigin:

$$f_4(w) = 10n + \sum_{i=0}^{n} w_i^2 - 10\cos(2\pi w_i) \quad (10)$$

posee un mínimo global en $f4(0, ..., 0) = 0$. Rastrigin es continua, diferenciable, separable, separable y multimodal.

5. Rosenbrock:

$$f_5(w) = \sum_{i=0}^{n-1}[100(w_{i+1} - w_i^2)^2 + (w_i - 1)^2] \quad (11)$$

posee un mínimo global en $f5(1, ..., 1) = 0$. Rosenbrock es continua, diferenciable, no-separable, no-separable y multimodal.

6. Schwefel:

$$f_6(w) = -1/n\sum_{i=0}^{n} w_i \sin\sqrt{|w_i|} \quad (12)$$

posee un mínimo global en $f6() = -418.983$. Schwefel es continua, diferenciable, separable, separable y multimodal.

7. Styblinski-Tang:

$$f_7(w) = 0.5 * \sum_{i=0}^{n} w_i^4 + 16w_i^2 + 5w_i \quad (13)$$

posee un mínimo global en $f7(-2.903534, \ldots, -2.903534) = -78.332$. Styblinski-Tang es continua, diferenciable, no-separable y multimodal.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

8. Step:

$$f_8(\boldsymbol{w}) = \sum_{i=0}^{n} (\lfloor \boldsymbol{w}_i + 0.5 \rfloor)^2 \qquad (14)$$

posee un mínimo global en $f8(0, ..., 0) = 0$. Step es discontinua, no-diferenciable, separable, separable y unimodal.

9. Alpine:

$$f_9(\boldsymbol{w}) = \sum_{i=0}^{n} |\boldsymbol{w}_i \sin(\boldsymbol{w}_i) + 0.1\boldsymbol{w}_i| \qquad (15)$$

posee un mínimo global en $f9(0, ..., 0) = 0$. Alpine es continua, no-diferenciable, separable, separable y unimodal.

Algoritmos meta-heurísticos de prueba

En el caso de los algoritmos de optimización, en el presente trabajo se consideran dos algoritmos meta-heurísticos para su comparación con GWO: *Particle Swarm Optimization (PSO)* y *Firefly Algorithm (FA).*

- Algoritmo PSO. El algoritmo PSO fue introducido en (Eberhart, 1995) y está basado en especies de animales, conceptualizados como partículas, cuyo movimiento coordinado expone algún tipo de experiencia o inteligencia de grupo en la consecución de determinado objetivo. La velocidad de una partícula se calcula según la **ecuación 16** donde $\omega 1$ y $\omega 2$ son dos vectores aleatorios de longitud n generados a partir de una distribución uniforme y $\odot$ es el producto de Hadamard. El vector w_i es la mejor partícula de la población de acuerdo al valor de la función objetivo y w_{i*} es la mejor posición visitada por la partícula i.

$$\boldsymbol{v}_i \leftarrow G\boldsymbol{v}_i + L\boldsymbol{\omega}_1 \odot \boldsymbol{w}_* + \gamma\boldsymbol{\omega}_2 \odot \boldsymbol{w}_{i*} \qquad (16)$$

  La **Ecuación 16** define tres híper-parámetros del algoritmo: α se conoce en la literatura como el factor de inercia, L mide la contribución del mejor individuo en la dirección de la partícula y γ mide la contribución de la mejor posición visitada en la dirección de la partícula. Una vez calculada la velocidad de la partícula, su movimiento viene dado por la **Ecuación 17.**

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \boldsymbol{v}_i \qquad (17)$$

  La variante PSO con acotación de velocidad propone calcular el híper-parámetro □a partir de dos híper-parámetros: Ma y Mi de forma tal que al inicio del algoritmo □$_0$= Ma. Posteriormente, en las siguientes iteraciones del algoritmo α decrece linealmente hasta Mi de la siguiente forma: □$_{i+1}$= 0.99□$_i$. La variante de

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

PSO con acotación de velocidad es un caso particular de la variante PSO con factor de restricción (Bonyadi, 2017).

El proceso de optimización mediante PSO se encuentra descrito en el Algoritmo PSO. En el paso 1 la población de partículas es inicializada aleatoriamente. Esto genera una cantidad nq de números aleatorios, siendo n la dimensión del vector de parámetros de cada individuo y q la cantidad de individuos.

Algoritmo 3.1: Algoritmo meta-heurístico de optimización PSO.

1: Inicializar aleatoriamente la población $\boldsymbol{w_i}$ siendo $i = 1, ..., q$
2: Establecer mejores posiciones visitadas de forma que $\boldsymbol{w_{i*}} \leftarrow \boldsymbol{w_i}$
3: Establecer mejor partícula global $\boldsymbol{w_*} \leftarrow \underset{\boldsymbol{w_i}}{\text{argmin}}(f(\boldsymbol{w_i}))$ para $1 \leq i < q$
4: **while** CONVERGENCIA == False **do**
5: **for** j = 1 : q **do**
6: **if** $f(\boldsymbol{w_j}) < f(\boldsymbol{w_{j*}})$ **then**
7: Actualizar $\boldsymbol{w_{j*}} \leftarrow \boldsymbol{w_j}$
8: **end if**
9: **if** $f(\boldsymbol{w_j}) < f(\boldsymbol{w_*})$ **then**
10: Actualizar $\boldsymbol{w_*} \leftarrow \boldsymbol{w_j}$
11: **end if**
12: **end for**
13: **for** j = 1 : q **do**
14: Calcular velocidad de la partícula j de acuerdo a la Ecuación 3.10
15: Calcular nueva posición de la partícula j de acuerdo a la Ecuación 3.11
16: **end for**
17: **end while**
18: **return** Partícula con el menor valor de la función objetivo $\boldsymbol{w_*}$

Figura 3. Pseudocódigo del algoritmo PSO

En los pasos 2 y 3 del algoritmo PSO se establecen los valores iniciales para las mejores posiciones visitadas y el mejor individuo local. En el paso 4 se comprueba que el algoritmo no se encuentre en un estado de convergencia. Para simplificar la descripción del algoritmo, se omite la definición del método $Convergencia(\eta)$ que devuelve verdadero si la cantidad de evaluaciones de la función objetivo es mayor que η o falso en caso contrario. Luego, en cada iteración se actualiza la mejor posición visitada de cada partícula en el paso 7 y la mejor posición global en el paso 10. Posteriormente se re-calcula la velocidad de cada partícula en el paso 14 y consecuentemente se actualiza su posición en el espacio de búsqueda en el paso 15. Finalmente se devuelve el individuo con menor valor de la función objetivo en el paso 18.

- Algoritmo FA. El algoritmo FA fue introducido en (Yang, 2009) y se inspira en el apareamiento de las luciérnagas mediante destellos de luz. La distancia entre dos luciérnagas usualmente se calcula mediante la

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

distancia euclidiana, **Ecuación 18**. De ahí que, el movimiento de una luciérnaga w_i hacia una luciérnaga w_j venga dado por la **Ecuación 19**.

$$r(\boldsymbol{w}_i, \boldsymbol{w}_j) = \sqrt{\sum_{k=1}^{n} (\boldsymbol{w}_{i,k} - \boldsymbol{w}_{j,k})^2} \tag{18}$$

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + Ibe^{-Ldr(\boldsymbol{w}_i, \boldsymbol{w}_j)^2}(\boldsymbol{w}_j - \boldsymbol{w}_i) + Ri\boldsymbol{\omega} \tag{19}$$

En la **Ecuación 19** se definen tres híper-parámetros: Ib el brillo inicial cuando la distancia entre dos luciérnagas es cero, Ld el coeficiente de atenuación de la luz y Ri la influencia de la aleatoriedad. En la ecuación, ω es un vector de longitud n generado aleatoriamente a partir de una distribución normal. La **figura 4** describe en pseudocódigo para el algoritmo FA.

Algoritmo 3.2: Algoritmo meta-heurístico de optimización FA.

```
1: Inicializar aleatoriamente la población w_i siendo i = 1, ..., q
2: Establecer mejor individuo w_* ← argmin_{w_i} f(w_i) para 1 ≤ i < q
3: while CONVERGENCIA == False do
4:     Ordenar los individuos de menor a mayor considerando la función objetivo
5:     for j = 1 : q do
6:         for k = 1 : q do
7:             if f(w_k) < f(w_j) then
8:                 Mover individuo w_j hacia w_k de acuerdo a la Ecuación 3.13
9:             end if
10:        end for
11:        if f(w_j) < f(w_*) then
12:            Actualizar w_* ← w_j
13:        end if
14:    end for
15: end while
16: return Individuo con el menor valor de la función objetivo w_*
```

Figura 4 Pseudocódigo del Algoritmo FA

En el paso 1 del algoritmo FA, la población de luciérnagas es inicializada aleatoriamente en el espacio de búsqueda y en el paso 2 se establece el mejor individuo global. En el paso 3 se comprueba que el algoritmo no se encuentra en un estado de convergencia. Para simplificar el algoritmo, se omite la definición del método

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

$Convergencia(\eta)$ que devuelve verdadero si la cantidad de evaluaciones de la función objetivo es mayor que Ri o falso en caso contrario.

En caso de que el algoritmo FA no se encuentre en un estado de convergencia, en el paso 4 la población es ordenada teniendo en cuenta el valor de la función objetivo. Esto se realiza para cambiar el orden en que las luciérnagas son comparadas. En cada iteración posterior, cada luciérnaga se mueve en cierta medida hacia cada una de las luciérnagas que posean mayor atracción en el paso 8 y en el paso 12 se actualiza el mejor individuo de la población. Finalmente, en el paso 16 se devuelve la luciérnaga con el menor valor de la función objetivo una vez que la cantidad máxima de iteraciones fue alcanzada.

Variables dependientes

Como variables dependientes se identifica la precisión y la eficiencia. Se entiende como precisión el valor de la función objetivo luego de que el algoritmo metaheurístico haya convergido en una solución. Como eficiencia se entiende el tiempo de ejecución luego de que el algoritmo metaheurístico haya convergido en una solución medido en milisegundos. En las pruebas realizadas se utilizó un procesador Core i3 con una frecuencia de 2.8 GHz y 4Gb de memoria RAM, ejecutando el sistema operativo Ubuntu 18.04.

Variables ajenas

Los híper-parámetros de los algoritmos metaheurísticos estudiados en el presente trabajo se consideran una variable ajena que afecta la precisión y eficiencia de los mismos. Por ello, se adopta un enfoque híbrido para su configuración. Para los híper-parámetros específicos de cada meta-heurística, especificados en el presente epígrafe, se emplea el algoritmo Tree Parzen Estimators (TPA) (Bergstra, 2011) con una cantidad de 100 iteraciones cuya implementación se encuentra disponible en Internet.

En adición a los híper-parámetros específicos de cada algoritmo meta-heurístico descrito, existe un conjunto de híper-parámetros generales comunes que afectan el comportamiento de los mismos. En el caso de estos híper-parámetros generales, se emplea la siguiente configuración manual para todos los algoritmos meta-heurísticos:

- Tamaño de la población: 40 individuos.
- Criterio de convergencia: 500 evaluaciones de la función objetivo.
- Espacio de búsqueda: -10 < w_i < 10
- Cantidad de dimensiones: 10

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

La cuestión más importante a considerar con las variables ajenas es controlarlas para evitar que su efecto sesgue los resultados de las pruebas de precisión y eficiencia. Es por esto que en el presente trabajo, se sigue rigurosamente el proceso de configuración de híper-parámetros para todos los algoritmos meta-heurísticos estudiados.

Configuración automática de los híper-parámetros

En el presente epígrafe se presentan los resultados de configurar los híper-parámetros de los algoritmos de optimización estudiados mediante el algoritmo TPE. En la **Tabla 1** se muestra para cada híper-parámetro de los algoritmos meta-heurísticos, su valor optimizado para cada función objetivo estudiada. De esta forma, se facilita la posibilidad de reproducir los resultados relacionados con la precisión y eficiencia del proceso de optimización que se presenta en los epígrafes posteriores. Note que, para el caso del algoritmo GWO no se muestra ningún valor debido a que este algoritmo no posee híper-parámetros.

Tabla 1. Híper-parámetros para cada algoritmo meta-heurístico de optimización y función objetivo estudiada. En el caso del algoritmo GWO no se muestra ningún valor debido a que este algoritmo no posee híper-parámetros.

	PSO				FA		
	G	L	Mi	Ma	Ld	Ib	Ri
De Jung	5.85E-01	5.47E-01	1.33E-01	6.29E-01	1.00E-01	3.54E-01	4.84E-01
Ackley	4.43E-01	8.82E-01	3.80E-01	6.70E-01	2.77E-01	2.72E-01	1.51E-01
Giewangk	7.62E-01	7.40E-01	5.04E-01	6.72E-01	8.29E-01	8.67E-01	5.44E-01
Rastrigin	7.18E-01	5.69E-01	3.61E-01	6.90E-01	8.37E-01	2.89E-01	1.89E-01
Rosenbrock	5.38E-01	7.93E-01	5.17E-01	7.01E-01	6.79E-01	5.41E-01	8.68E-01
Schwefel	7.92E-01	6.07E-01	4.29E-01	6.67E-01	3.18E-01	9.00E-01	1.27E-01
Styblinski...	7.65E-01	6.86E-01	3.73E-01	6.11E-01	3.52E-01	1.06E-01	2.79E-01
Step	8.55E-01	3.45E-01	2.39E-01	8.54E-01	4.23E-01	8.42E-01	2.68E-01
Alpine	7.98E-01	7.64E-01	5.63E-01	6.93E-01	7.37E-01	8.74E-01	4.39E-01

Pruebas de precisión

En la **Figura 5** se muestran los resultados obtenidos considerando la precisión de los distintos algoritmos de optimización respecto a cada función objetivo estudiada en 10 mediciones. En el eje horizontal se grafican los algoritmos utilizados en la optimización y en el eje vertical el valor de la función objetivo. Como se trata de problemas de minimización, mientras más elevado sea el valor de la función objetivo menor precisión. La parte inferior y superior de las cajas representan el primer y tercer cuartil respectivamente. La línea divisoria indica la mediana de las mediciones. Los brazos de las cajas representan la desviación estándar.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

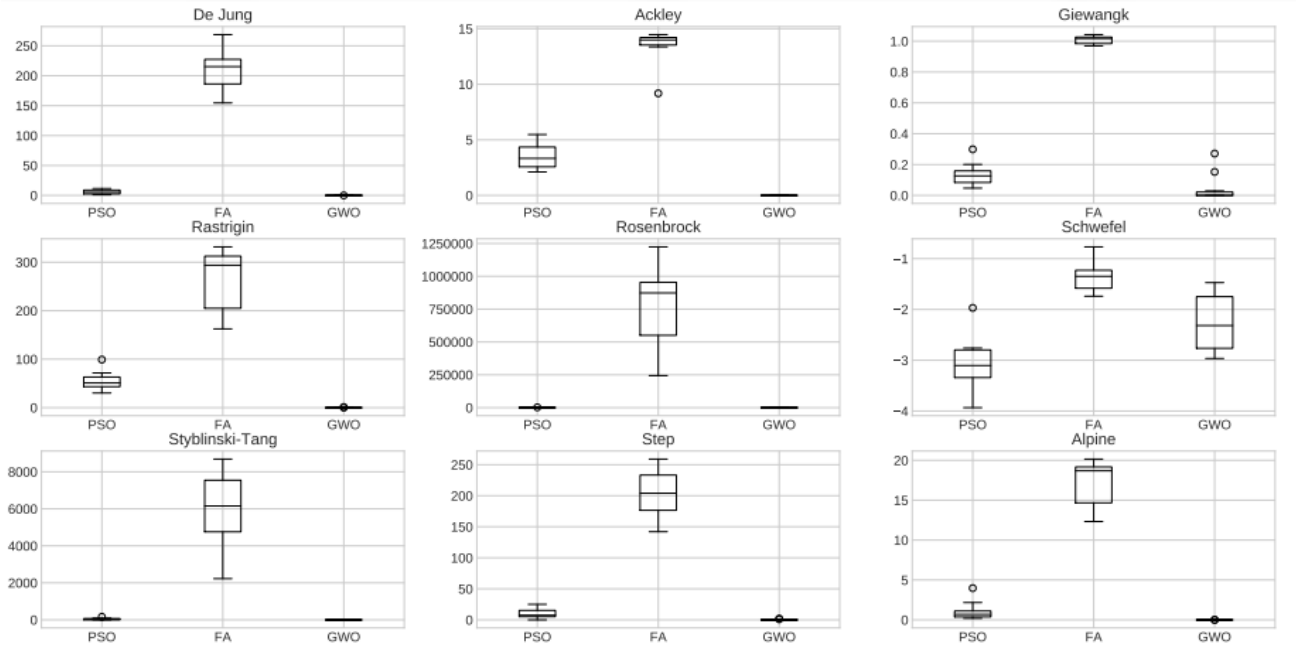

Figura 5. Precisión del proceso de optimización utilizando diferentes algoritmos meta-heurísticos aplicados a la optimización de diferentes funciones objetivos.

En la **Tabla 2** se presenta un resumen de los resultados mostrados en la **Figura 5**. Para cada algoritmo meta-heurístico y función objetivo se muestra el promedio de 10 mediciones realizadas.

Tabla 2. Precisión del proceso de optimización utilizando diferentes algoritmos meta-heurísticos aplicados a la optimización de diferentes funciones objetivos.

	PSO	FA	GWO
Ackley	3.49E+00	1.35E+01	2.21E-05
Alpine	1.09E+00	1.72E+01	2.36E-04
De Jung	5.73E+00	2.09E+02	3.85E-10
Giewangk	1.36E-01	1.01E+00	4.53E-02
Rastrigin	5.56E+01	2.65E+02	1.28E-01
Rosenbrock	5.49E+02	7.62E+05	8.45E+00
Schwefel	-3.07E+00	-1.34E+00	-2.24E+00
Step	1.00E+01	2.04E+02	3.00E-01
Styblinski-Tang	4.61E+01	5.88E+03	-1.14E+00

Como puede apreciarse, los resultados alcanzados muestran que en una cantidad importante de funciones objetivos, el algoritmo GWO obtiene una precisión superior al ser comparado con los algoritmos meta-heurísticos PSO y FA. Esto

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

resulta especialmente notorio, al considerar que el algoritmo GWO no posee híper-parámetros ajustables como PSO y FA. Entre los tres algoritmos estudiados, el algoritmo FA es el que obtiene menor precisión, obteniendo los valores más elevados de la función objetivo. A pesar de los resultados positivos, la precisión obtenida por el algoritmo GWO no se corresponde con la precisión reportada en el trabajo original donde fue introducido (Mirjalili, 2014).

Pruebas de eficiencia

En la **Figura 6** se muestran los resultados obtenidos considerando la eficiencia de los distintos algoritmos de optimización respecto a cada función objetivo estudiada en 10 mediciones. En el eje horizontal se grafican los algoritmos utilizados en la optimización y en el eje vertical el valor del tiempo de ejecución. Mientras más elevado sea el valor del tiempo de ejecución menor eficiencia. Como antes, la parte inferior y superior de las cajas representan el primer y tercer cuartil respectivamente. La línea divisoria indica la mediana de las mediciones. Los brazos de las cajas representan la desviación estándar.

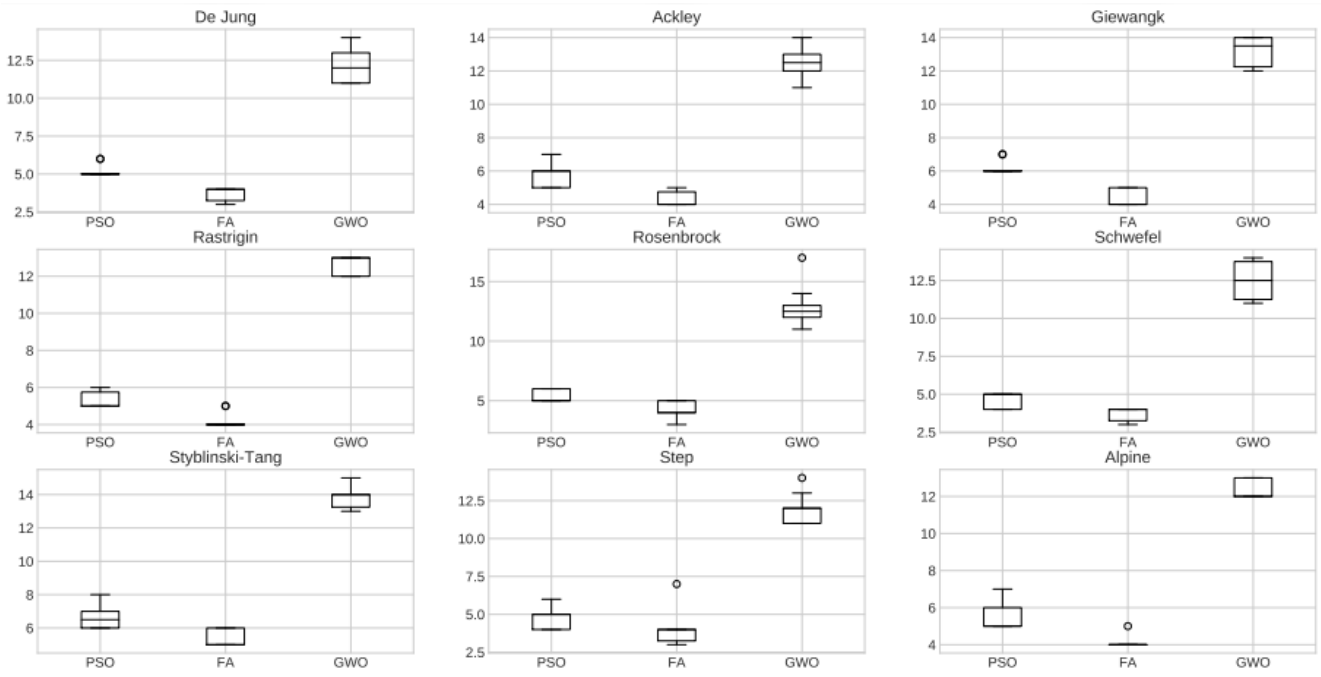

Figura 6. Eficiencia del proceso de optimización utilizando diferentes algoritmos meta-heurísticos aplicados a la optimización de diferentes funciones objetivos.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 8, Mes Agosto, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-194
http://publicaciones.uci.cu

En la **Tabla 3** se presenta un resumen de los resultados mostrados en la **Figura 6**. Para cada algoritmo meta-heurístico y función objetivo se muestra el promedio de 10 mediciones realizadas.

Tabla 3. Eficiencia del proceso de optimización utilizando diferentes algoritmos meta-heurísticos aplicados a la optimización de diferentes funciones objetivos.

	PSO	FA	GWO
Ackley	5.80E+00	4.30E+00	1.25E+01
Alpine	5.50E+00	4.10E+00	1.24E+01
De Jung	5.20E+00	3.70E+00	1.21E+01
Giewangk	6.20E+00	4.40E+00	1.32E+01
Rastrigin	5.30E+00	4.20E+00	1.26E+01
Rosenbrock	5.40E+00	4.30E+00	1.29E+01
Schwefel	4.60E+00	3.70E+00	1.25E+01
Step	4.80E+00	4.00E+00	1.19E+01
Styblinski-Tang	6.70E+00	5.40E+00	1.39E+01

Como puede apreciarse, los resultados alcanzados muestran que en una cantidad importante de funciones objetivos, el algoritmo GWO es menos eficiente al ser comparado con los algoritmos meta-heurísticos PSO y FA. No obstante, la diferencia no es notoria al tratarse de una diferencia en el orden de los milisegundos. No obstante, puede esperarse que en problemas con una mayor cantidad de dimensiones o al realizar una mayor cantidad de iteraciones de los algoritmos, esta diferencia sea superior.

Conclusiones

GWO es menos eficiente que los algoritmos PSO y FA, no obstante, la diferencia observada en tiempo de ejecución se encuentra en el orden de los milisegundos.

El algoritmo GWO obtiene mayor precisión al ser comparado con los algoritmos PSO y FA a pesar de no poseer híper-parámetros ajustables a cada problema de optimización.

A pesar de los resultados positivos, la precisión obtenida por el algoritmo GWO no se corresponde con la precisión reportada en el trabajo original donde fue introducido.

Referencias

GIGERENZER, Gerd; BRIGHTON, Henry. Homo heuristicus: Why biased minds make better inferences. Topics in cognitive science, 2009, vol. 1, no 1, p. 107-143.

CLANCEY, William J. Problem in the Design of Intelligent Machines. Architectures for intelligence, 1991, p. 357.

SORENSEN, Kenneth; SEVAUX, Marc; GLOVER, Fred. A history of metaheuristics. arXiv preprint arXiv:1704.00853, 2017.

ZHENG, Bichen, et al. Predictive modeling of hospital readmissions using metaheuristics and data mining. Expert Systems with Applications, 2015, vol. 42, no 20, p. 7110-7120.

JOSÉ-GARCÍA, Adán; GÓMEZ-FLORES, Wilfrido. Automatic clustering using nature-inspired metaheuristics: A survey. Applied Soft Computing, 2016, vol. 41, p. 192-213.

MESEJO, Pablo, et al. Biomedical image segmentation using geometric deformable models and metaheuristics. Computerized Medical Imaging and Graphics, 2015, vol. 43, p. 167-178.

ZHENG, Yu-Jun. Water wave optimization: a new nature-inspired metaheuristic. Computers & Operations Research, 2015, vol. 55, p. 1-11.

OSMAN, Ibrahim H.; KELLY, James Patrick. Meta-heuristics theory and applications. Journal of the Operational Research Society, 1997, vol. 48, no 6, p. 657-657.

VOß, Stefan. Meta-heuristics: The state of the art. En Workshop on Local Search for Planning and Scheduling. Springer, Berlin, Heidelberg, 2000. p. 1-23.

MIRJALILI, Seyedali; MIRJALILI, Seyed Mohammad; LEWIS, Andrew. Grey wolf optimizer. Advances in engineering software, 2014, vol. 69, p. 46-61.

LIANG, Jane-Jing; SUGANTHAN, Ponnuthurai Nagaratnam; DEB, Kalyanmoy. Novel composition test functions for numerical global optimization. En Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005. IEEE, 2005. p. 68-75.

JAMIL, Momin; YANG, Xin-She. A literature survey of benchmark functions for global optimisation problems. International Journal of Mathematical Modelling and Numerical Optimisation, 2013, vol. 4, no 2, p. 150-194.

EBERHART, Russell; KENNEDY, James. A new optimizer using particle swarm theory. En MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Ieee, 1995. p. 39-43.

BONYADI, Mohammad Reza; MICHALEWICZ, Zbigniew. Particle swarm optimization for single objective continuous space problems: a review. 2017.

YANG, Xin-She. Firefly algorithms for multimodal optimization. En International symposium on stochastic algorithms. Springer, Berlin, Heidelberg, 2009. p. 169-178.

BERGSTRA, James S., et al. Algorithms for hyper-parameter optimization. En Advances in neural information processing systems. 2011. p. 2546-2554.