Análisis Comparativo de los Algoritmos Basados en Abejas y Hormigas en el Problema de la Esfera

Hernández-Ocaña, Betania [1]; **Hernández-Torruco, José** [1]; **Chávez-Bosquez, Oscar** [1,*]; **Montané-Jiménez, Luis G.** [2]

[1] *Universidad Juárez Autónoma de Tabasco, División Académica de Ciencias y Tecnologías de la Información, Cunduacán, Tabasco, México*
[2] *Universidad Veracruzana, Facultad de Estadística e Informática, Xalapa de Enríquez, Veracruz, México*

Resumen: Dos algoritmos bio-inspirados en la naturaleza se implementaron con el fin de conocer y analizar su comportamiento al buscar una solución a un problema de optimización numérica bien conocido en el estado del arte como problema de la esfera. Asimismo, se buscó detectar aspectos comunes y particulares de ambos algoritmos que derivan en una convergencia prematura. Estos algoritmos son: el Algoritmo de Optimización de Colonia de Hormigas (ACO) y Colonia Artificial de Abejas (ABC). Ambos pertenecen al grupo de Algoritmos de Inteligencia Colectiva, los cuales simulan el comportamiento colaborativo de ciertas especies simples e inteligentes. ACO y ABC son escasamente utilizados debido a su poca popularidad en la solución de problemas de optimización numérica puesto que en un principio fueron desarrollados para problemas combinatorios. Los resultados de cinco experimentos son presentados dando como mejor algoritmo a ACO para el problema de la esfera.

Palabras claves: Optimización, Colonia de Hormigas, Colonia Artificial de Abejas, Metaheurísticas.

A Comparative Study of Bee and Ant Algorithms on the Sphere Problem

Abstract: Two bio-inspired algorithms in nature were implemented in order to know and analyze their behavior when looking for a solution to a numerical optimization problem well-known in the state of the art as a sphere problem. Likewise, we sought to detect common and particular aspects of both algorithms that lead to a premature convergence. These algorithms are: the Ant Colony Optimization Algorithm (ACO) and the Artificial Bee Colony (ABC). Both belong to the group of Collective Intelligence Algorithms, which simulate the collaborative behavior of certain simple and intelligent species. ACO and ABC are rarely used due to their low popularity in solving numerical optimization problems since they were originally developed for combinatorial problems. The results of five experiments are presented giving ACO as the best algorithm for the sphere problem.

Keywords: Optimization, Ant Colony, Artificial Bee Colony, Metaheuristics.

1. Introducción

Los algoritmos bio-inspirados surgen con la motivación de mejorar algoritmos de búsqueda para resolver problemas de optimización. Los algoritmos bio-inspirados son usados como metaheurísticas debido a que arrojan un conjunto de resultados a un problema en particular de manera total o aproximada al óptimo global. De acuerdo al fenómeno natural en que basan su diseño, se clasifican en dos grupos: Los Algoritmos Evolutivos (AEs) que emulan el proceso evolutivo de las especies (Eiben and Smith, 2003) y los Algoritmos de Inteligencia Colectiva (AICs) que emulan el comportamiento colaborativo de ciertas especies (algunas de ellas simples y otras inteligentes) como bacterias, abejas, hormigas, aves, peces, monos, entre otros (Engelbrecht, 2005).

En las últimas cuatro décadas, el uso de metaheurísticas ha planteado ser una alternativa eficaz para resolver Problemas de Optimización Numérica con Restricciones (PONR). Aunque en un principio estos algoritmos fueron creados para resolver problemas de optimización sin restricciones, se han creado mecanismos adicionales para el manejo de restricciones tales como: función de penalización, decodificadores, operadores especiales, separación de la función objetivo y restricciones, reglas de factibilidad, entre otras (Mezura-Montes and Coello-Coello, 2011).

Un PONR también es conocido como el problema general de programación no-lineal y se puede definir como:

$$\begin{gathered}\text{Minimizar } f(\vec{x}) \text{ sujeta a:}\\ g_i(\vec{x}) \leq 0,\ i = 1,\ldots,m,\\ h_j(\vec{x}) = 0,\ j = 1,\ldots,p,\end{gathered}$$

*oscar.chavez@ujat.mx
Recibido: 29/09/2020
Aceptado: 21/06/2022
Publicado: 04/08/2022
10.33333/rp.vol50n2.06

Donde $\vec{x} \in R^n$ tal que $n \geq 1$, es el vector de soluciones $\vec{x} = [x_1, x_2, \ldots, x_n]^T$, donde cada x_i, $i = 1, \ldots, n$ está delimitada por el límite inferior y superior $L_i \leq x_i \leq U_i$; m es el número de restricciones de desigualdad y p es el número de restricciones de igualdad (en ambos casos, las restricciones podrían ser lineales o no lineales). Si denotamos con F a la región factible (donde se encuentran todas las soluciones que satisfacen al problema) y con S a todo el espacio de búsqueda, entonces debe ser claro que $F \subseteq S$ Hernández-Ocaña et al. (2016).

Los AEs son ampliamente utilizados como técnicas alternativas para resolver PONR. Los principales paradigmas son: los Algoritmos Genéticos (AG), la Programación Genética (PG), las Estrategias Evolutivas (EE), la Programación Evolutiva (PE), y la Evolución Diferencial(ED). Por otro lado, el AIC más popular es Optimización Mediante Cúmulos de Partículas (PSO, por sus siglas en inglés) (Kennedy and Eberhart, 2001). Sin embargo, otras estrategias relativamente recientes han llamado la atención en el área por su utilidad para resolver con éxito problemas de optimización: Algoritmo de Optimización de Colonia de Hormigas (ACO, por las siglas en inglés de *Ant Colony Optimization*) (Dorigo et al., 1996) y Colonia Artificial de Abejas (ABC, por las siglas en inglés de *Artificial Bee Colony*) (Karaboga and Basturk, 2007), ambos algoritmos principalmente se han utilizado para resolver problemas combinatorios (Abbass, 2001; Leguizamón and Coello-Coello, 2009; Haddad and Mariño, 2007; Baykasoglu et al., 2007).

ACO simula el comportamiento de las hormigas en busca de comida y su regreso al nido. Al inicio, la hormiga busca una fuente alimentaria. Si la encuentra, la hormiga deja un rastro de feromona a su regreso al nido, la cual atrae a las hormigas cercanas, de modo que sigan este rastro. Si más hormigas utilizan el mismo camino encontrado entre el nido y la fuente alimentaria, éste será escogido como el camino más corto por la alta concentración de feromonas.

En las últimas cuatro décadas, el uso de metaheurísticas ha planteado ser una alternativa eficaz para resolver Problemas de Optimización Numérica con Restricciones (PONR). Aunque en un principio, estos algoritmos fueron creados para resolver problemas de optimización sin restricciones, se han creado mecanismos adicionales para el manejo de restricciones tales como: función de penalización, decodificadores, operadores especiales, separación de la función objetivo y restricciones, reglas de factibilidad, entre otras (Mezura-Montes and Coello-Coello, 2011).

ABC simula el comportamiento de las abejas en la búsqueda de alimento. Biológicamente, las abejas realizan la búsqueda de fuentes de alimento a grandes distancias y al regresar a su panal revolotean haciendo danzas, las cuales transmiten la información de las fuentes de alimento localizadas. De manera artificial, el objetivo es el reclutamiento de abejas en una fuente de alimento y abandono de la misma (Karaboga and Basturk, 2007).

ACO y ABC son algoritmos cuyo potencial aún no es aprovechado a su máximo debido a su poca popularidad (Mezura-Montes and Cetina-Domínguez, 2012; Dorzán et al., 2012; Leguizamón and Coello-Coello, 2009), aunque existen soluciones que han implementado estas metaheurísticas con éxito en problemas del mundo real (Matijaš et al., 2010; Vega-Alvarado et al., 2018). Por ello, esta investigación es motivada principalmente para explorar las capacidades de ambos algoritmos en la solución de un problema de optimización numérica conocido en el estado del arte como la minimización de una esfera.

Este problema es probado en 5 experimentos, donde se prueba el problema con dos y diez dimensiones, lo cual hace más complejo al problema. El objetivo principal de estos experimentos es detectar las características particulares y comunes que permiten un mejor rendimiento de estas dos metaheurísticas. Se busca analizar el beneficio de incrementar o decrementar las poblaciones de abejas y hormigas; así mismo de las iteraciones generacionales de estos algoritmos. Ambos algoritmos serán probados con los mismos parámetros para hacer una comparación justa al resolver el mismo problema de optimización.

El presente trabajo está organizado de la siguiente manera: En la Sección 1 se describen brevemente los antecedentes de esta investigación dando una introducción a los algoritmos bio-inspirados, presentación del modelado del Problema de Optimización Numérica con Restricciones, y de los algoritmos a implementar en este trabajo: Optimización de Colonia de Hormigas y Colonia Artificial de Abejas. En la Sección 2, se describe brevemente al algoritmo de la Colonia Artificial de Abejas. De igual forma, el Algoritmo de Optimización de Colonia de Hormigas es descrito en la Sección 3. Los resultados de implementar ambos algoritmos en el problema de optimización de la esfera son presentados en el Sección 4. Finalmente, las conclusiones y trabajos futuros son presentados en la Sección 5.

2. Colonia Artificial de Abejas (ABC)

ABC se compone de 3 grupos de abejas que trabajan durante un máximo número de iteraciones *MCN* del algoritmo: abejas empleadas, abejas observadoras y abejas exploradoras. El número de abejas empleadas es usualmente igual al número de fuentes de alimento *SN* y se asigna una abeja empleada a cada una de las fuentes. Al llegar a dicha fuente, la abeja calculará una nueva solución (volará hacia otra fuente de alimento cercana) a partir de esta y conservará la mejor solución. El número de abejas observadoras es usualmente igual al número de abejas empleadas y son creadas aleatoriamente dentro del rango del espacio de búsqueda, sin embargo, se puede definir un parámetro para determinar el número de abejas observadoras. En este trabajo no se define otro parámetro. La función de la abeja observadora es visitar una fuente de alimento asignada con base en la aptitud de las abejas empleadas (valor de la función objetivo) y se calcula una nueva solución a partir de su fuente de alimento asignada y la información de la propia abeja observadora. Cuando una fuente de alimento no mejora después de un cierto número de iteraciones definido con el parámetro *limit*, ésta se abandona, siendo reemplazada por aquella encontrada por una abeja exploradora, la cual es una nueva abeja generada aleatoriamente Mezura-Montes et al. (2010).

Tabla 1. Parámetros del ABC

Símbolo	Descripción
SN	Número de soluciones (fuentes de alimento).
MCN	Número total de iteraciones que ejecutará el ABC.
limit	Número de iteraciones que se conservará una solución sin mejora antes de ser reemplazada por una nueva generada por una abeja exploradora.

La ventaja de este algoritmo es el bajo número de parámetros que se requiere calibrar, como se puede observar en la Tabla 1.

La representación de las soluciones se lleva a cabo mediante fuentes de alimentos, los cuales son vectores de D-dimensiones (donde D es el número de variables de decisión del problema). La representación vectorial de la solución i en la iteración g es la presentada en la Ecuación (1):

$$x_{i,g}, i = 1,\ldots,SN, \quad g = 1,\ldots,MCN \tag{1}$$

donde cada variable de decisión x_i está asociada a un rango $L_i \leq x_i \leq U_i$, el cual se debe tomar en cuenta para generar las fuentes de alimento iniciales de manera aleatoria con una distribución uniforme.

En el mecanismo de selección, la comunicación entre abejas empleadas y observadoras es indispensable, de manera que aquellas fuentes de comida con mejor calidad (mejor valor de aptitud) serán más visitadas. Las abejas son vistas como operadores de variación, cuando una de ellas llega a una fuente de alimento, se calcula una nueva solución candidata $v_{i,g}$ utilizando la Ecuación (2):

$$v_{i,g} = x_{i,g} + \phi * (x_{i,g} - x_{k,g}) \tag{2}$$

donde $x_{i,g}$ representa la opción actual de la abeja, $x_{k,g}$ es una fuente de alimento aleatoria y distinta de $x_{i,g}$, g es la iteración actual del algoritmo y ϕ es un número real aleatorio en el intervalo [-1,1].

El mecanismo para eliminar soluciones (fuentes de comida) es mediante las abejas exploradoras, es decir, cuando una solución no es mejorada (no es reemplazada por una candidata) durante el número de iteraciones establecidos por la variable *limit* donde su valor es determinado por $(SN * D)$. La fuente abandonada se reemplaza con una nueva solución generada aleatoriamente. Finalmente, la mejor solución encontrada durante la iteración se compara con la mejor solución en memoria y si tiene una mejor aptitud la reemplazará. El pseudocódigo de ABC es presentado en el Algoritmo 1.

Algoritmo 1: Pseudocódigo de ABC

```
begin
    Crear una población inicial de abejas x_{i,0}, i = 1,...,SN
    Evaluar x_{i,0}, i = 1,...,SN
    g=1
    repeat
        Producir nuevas soluciones v_{i,g} para las abejas empleadas
          utilizando la Ecuación (2) y evaluarlas.
        Conservar la mejor solución entre la actual y la candidata.
        Seleccionar las soluciones que serán visitadas por una abeja
          observadora según el valor de aptitud.
        Producir nuevas soluciones v_{i,g} para las abejas observadoras
          utilizando la Ecuación (2) y evaluarlas.
        Conservar la mejor solución entre la actual y la candidata.
        Determinar si existe una fuente abandonada y reemplazarla
          utilizando una abeja exploradora.
        Memorizar la mejor solución encontrada hasta el momento.
        g=g+1
    until g==MCN;
end
```

3. Colonia de Hormigas (ACO)

ACO se inspira en el comportamiento de búsqueda y provisión de alimentos de algunas especies de hormigas. Estas hormigas depositan feromona en el suelo con el fin de marcar algún camino favorable que debe ser seguido por otros miembros de la colonia. Los parámetros de ACO se presentan en la Tabla 2.

Como prerrequisito para aplicar un algoritmo ACO, se requiere de un "grafo de construcción". La existencia de este grafo permitirá a las hormigas de la colonia recorrer dicho grafo para la construcción de las soluciones en forma cooperativa. Las hormigas artificiales construyen una solución para atravesar el grafo de construcción $G_C(V,E)$. Este grafo totalmente conectado consiste de un conjunto de vértices V y un conjunto de arcos E. El conjunto de componentes C puede ser asociado con el conjunto de vértices V, o con el conjunto de arcos E. Las hormigas se mueven entre los vértices a lo largo de los arcos del grafo, construyendo incrementalmente una solución parcial. Además, las hormigas depositan una cierta cantidad de feromona sobre los componentes, es decir, en los vértices o en los arcos que atraviesan. La cantidad de feromona $\Delta\tau$ depositada puede depender de la calidad de la solución encontrada. Las hormigas siguientes utilizan la información de la feromona como una guía hacia regiones más prometedoras del espacio de búsqueda. El pseudocódigo de ACO es presentado en el Algoritmo 2 donde la aplicación del buscador local será decisión del usuario (La función `AplicarBúsquedaLocal()` es opcional).

En ACO una variable de decisión instanciada $X_i = v_i^j$ (es decir, una variable X_i con un valor v_i^j asignado de su dominio D_i, donde D es el dominio de las variables $D_1,\ldots,D_n$ y v_i^j es una distancia del grafo), se llama componente de una solución y se denota por c_{ij}. El conjunto de todos las posibles componentes de soluciones es denotado por C. Un parámetro de rastro de feromona T_{ij} es asociado con cada componente c_{ij}. El conjunto de todos los parámetros de rastros de feromona es denotado por T. El valor de un parámetro de rastro de feromona T_{ij} es denotado por τ_{ij} (y es llamado valor de feromona). Este valor de feromona es usado y actualizado por el algoritmo ACO durante la búsqueda, y permite modelar la distribución de probabilidad de diferentes componentes de una solución (Arito, 2010).

`ConstruirSolucionesporHormigas()` Un conjunto de m hormigas artificiales construye soluciones a partir de elementos de un conjunto finito de componentes de soluciones disponibles $C = c_{ij}, i = 1,\ldots,n,\ j = 1,\ldots,|D_i|$. La construcción de una so-

Algoritmo 2: Pseudocódigo de ACO

```
1  x_i, 0, i = 1, ..., h
2  Generar grafo de solución para cada x_i, 0, i = 1, ..., h.
3  N = 1
4  begin
5      while N ≤ Nro_iters do
6          ConstruirSolucionesporHormigas() de acuerdo
           con la Ecuación (3)
7          [AplicarBúsquedaLocal()] (por ejemplo:
           Búsqueda Tabú, Programación Cuadrática Secuencial,
           entre otros).
8          ActualizarFeromona() de acuerdo con la
           Ecuación (4).
9      end
10 end
```

lución empieza con una solución parcial vacía $s^p = 0$. Luego, en cada paso de la construcción, la solución parcial actual s^p es extendida agregando una componente de solución factible del conjunto de vecinos factibles $N(s^p) \subseteq C$. El proceso de construcción de soluciones puede ser considerado como un camino en el grafo de construcción $G_C(V,E)$. Los caminos permitidos en G_C están definidos implícitamente por el mecanismo de construcción de soluciones que define el conjunto $N(s^p)$ con respecto a una solución parcial s^p. La elección de una componente de solución de $N(s^p)$ se hace probabilísticamente en cada paso de la construcción. Las reglas exactas para la elección probabilística de componentes de soluciones varían entre diferentes variantes de algoritmos ACO. La regla más conocida es de *Ant System* (Dorigo et al., 1996) y se presenta en la Ecuación (3).

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} * \eta(c_{ij})^{\beta}}{\sum c_{ij} \in N(s^p) \tau_{ij}^{\eta} * \eta(c_{ij})^{\beta}}, \forall c_{ij} \in N(s^p) \quad (3)$$

donde τ_{ij} es el valor de feromona asociado con la componente c_{ij} , y $\eta(.)$ es una función que asigna a cada paso de la construcción un valor heurístico para cada componente de solución factible $c_{ij} \in N(s^p)$. Los valores que son retornados por esta función son llamados normalmente información heurística. Además, α y β son parámetros aleatorios positivos entre [0,1] cuyos valores determinan la importancia relativa de la feromona y de la información heurística, respectivamente.

`ActualizarFeromona()` El objetivo de la actualización de feromona es incrementar los valores de feromona asociados con soluciones buenas o prometedoras, y reducir aquellos valores que están asociados con malas soluciones. Normalmente, esto se logra: 1) reduciendo todos los valores de feromona a través de la evaporación de feromona (evitar convergencia prematura del algoritmo) y 2) aumentando los niveles de feromona asociados con un conjunto de buenas soluciones S_{upd} como se calcula con la Ecuación (4).

Tabla 2. Parámetros de ACO

Símbolo	Descripción
h	Tamaño de la población de hormigas.
Nro_iters	Número total de iteraciones que ejecutará el ACO.
ρ	Factor de evaporación.

Tabla 3. Combinación de parámetros para la calibración de ACO y ABC

Algoritmo	Población	Iteración
ACO, ABC con 2 y 10 dimensiones	50	1000
ACO, ABC con 2 y 10 dimensiones	50	100
ACO, ABC con 2 y 10 dimensiones	20	1000
ACO, ABC con 2 y 10 dimensiones	20	100

$$\tau_{ij} \longleftarrow (1-\rho) * \tau_{ij} + \rho * \sum_{s \in S_{upd} | c_{ij} \in s} F(s) \quad (4)$$

donde S_{upd} es el conjunto de soluciones que son usadas para la actualización, $\rho \in (0,1]$ es un parámetro llamado factor de evaporación, y $F : S \longrightarrow R_0^+$ es una función tal que $f(s) < f(s') \Rightarrow F(s) \geq F(s'), \forall s \neq s' \in S$. La función $F(\dot{})$ es llamada función *fitness* o de aptitud.

S_{upd} es un subconjunto de $S_{iter} \cup s_{bs}$ donde S_{iter} es el conjunto de soluciones que fueron construidas en la iteración actual, y s_{bs} es la mejor solución encontrada desde la primera iteración del algoritmo. Un ejemplo muy conocido es la regla de actualización de *Ant System* $S_{upd} \leftarrow S_{iter}$.

4. Pruebas y resultados

Los algoritmos ACO y ABC fueron implementados en el lenguaje M usando el software libre Octave 4.2.1, el cual es una variante libre de Matlab (Eaton, 2016). El problema de optimización (minimización) a resolver por ambos algoritmos es la bien conocida función de la esfera, la cual es definida en la Ecuación (5).

$$f(x_1,x_2) = x_1^2 + x_2^2, -10 \leq x_1, x_2 \leq 10 \quad (5)$$

Se realizaron 5 experimentos para observar y analizar el comportamiento de los algoritmos ACO y ABC ante el problema de optimización numérica. Los primeros 4 experimentos consistieron en probar a ACO y ABC en el problema de la esfera con dos y 10 dimensiones empleando la configuración de parámetros mostrada en la Tabla 3 en un total de 10 ejecuciones independientes por cada algoritmo. El parámetro ρ fue un valor aleatorio entre (0,1] y *limit* fue $SN * D$ para todos los experimentos.

Para el caso de dos dimensiones, la función de la esfera se representa como una curva en el plano. En el caso de tres dimensiones, se representa como una esfera propiamente dicha. Para el caso de cuatro o más dimensiones corresponde al hiperespacio.

Los resultados de la ejecución independiente de los algoritmos con cada una de las combinaciones se presentan en la Tabla 4 de manera resumida, usando estadística básica como mejor y peor resultado, media, desviación estándar. Como se puede observar, en la mayoría de las ejecuciones el algoritmo ACO obtuvo mejores resultados con respecto a ABC y con una diferencia significativa según la prueba de Wilcoxon *Signed Rank Test*.

Para ambos algoritmos, resolver el problema de la esfera con diez dimensiones se vuelve más complejo que con sólo dos dimensiones. Por otra parte, se puede apreciar en la Tabla 4 que no importa el número de población que tenga el algoritmo ACO:

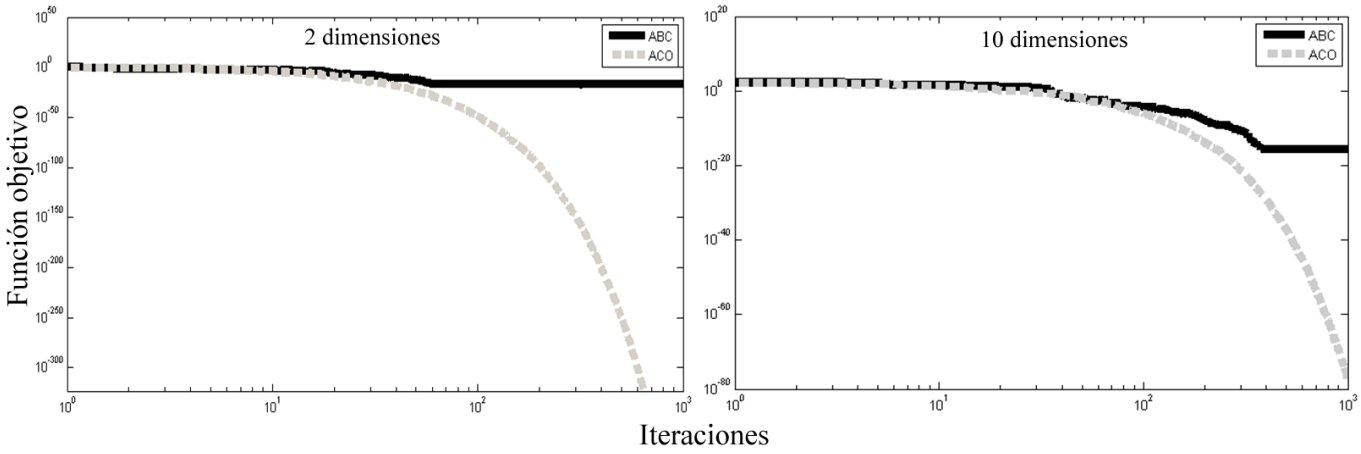

Figura 1. Gráfica de convergencia de ACO y ABC con 20 individuos y 1000 iteraciones en el problema de la esfera con 2 y 10 dimensiones

mientras más iteraciones haga, mejor será el resultado arrojado, puesto que se probó con población de 50 y 20 e iteraciones de 100 y 1000 obteniendo los mejores resultados con una población de 20 y 1000 iteraciones. Caso contrario ocurre con el algoritmo ABC, ya que a menor número de iteraciones se obtienen mejores resultados.

En general, ACO obtiene mejores resultados al tener un mayor número de iteraciones y menor número de población. Es decir, entre menos hormigas y más tiempo, se acarrea más comida. En el caso de ABC, entre menos tiempo y más abejas, se recolecta más miel. Se puede apreciar que para ACO, la mejor combinación de parámetros es la de una población de 20 y 1000 iteraciones; para ABC la mejor combinación de parámetros es población de 50 y 100 iteraciones, tomando como referencia al problema con una dimensionalidad mayor, es decir, el problema de la esfera con diez dimensiones.

Tabla 4. Resultados obtenidos por los algoritmos ACO y ABC en las diez ejecuciones con distinta combinación de parámetros

	2 dimensiones		**10 dimensiones**	
Estadística	**ACO**	**ABC**	**ACO**	**ABC**
	50 individuos y 1000 iteraciones			
Mejor	1.12E-279	2.65E-18	1.16E-37	1.27E-05
Media	2.37E-275	2.50E-17	2.91E-35	4.84E-05
Desv. estándar	0	1.61E-17	6.34E-35	3.22E-05
Peor	1.61E-274	4.64E-17	2.08E-34	9.86E-05
	50 individuos y 100 iteraciones			
Mejor	2.92E-29	1.82E-18	1.07E-02	1.08E-16
Media	4.60E-28	2.74E-18	2.94E-02	1.10E-16
Desv. estándar	5.97E-28	2.39E-18	1.40E-02	5.11E-17
Peor	2.02E-27	5.17E-18	4.94E-02	2.34E-16
	20 individuos y 1000 iteraciones			
Mejor	0.00E+00	4.53E-19	1.87E-79	3.10E-05
Media	0.00E+00	4.45E-18	3.31E-77	4.50E-04
Desv. estándar	0	5.28E-18	5.19E-77	2.83E-04
Peor	0.00E+00	1.63E-17	1.69E-76	9.40E-04
	20 individuos y 100 iteraciones			
Mejor	1.60E-52	4.20E-18	3.41E-07	3.74E-04
Media	1.52E-50	3.16E-17	4.40E-06	4.50E-04
Desv. estándar	1.69E-50	2.53E-17	5.56E-06	2.83E-04
Peor	5.24E-50	8.22E-17	1.64E-05	9.40E-04

Tabla 5. Parámetros de ACO y ABC para la ejecución de las 30 ejecuciones independientes

ACO		**ABC**	
Nro_iters	1000	*MCN*	1000
h	20	*SN*	20
ρ	$(0,1]$	*limit*	$SN*D$

En el quinto y último experimento, ACO y ABC son ejecutados en el problema de la esfera con la mejor combinación de parámetros de los experimentos previos. Ambos algoritmos se ejecutaron de manera independiente 30 veces. Los parámetros utilizados en los algoritmos ABC y ACO son presentados en la Tabla 5. Cabe mencionar que se usó el mismo número de iteraciones y población para observar la convergencia de los algoritmos de manera equitativa.

Los resultados de la ejecución de ambos algoritmos en la función de la esfera con dos y diez dimensiones se presentan en la Tabla 6. Para el caso de dos dimensiones, se observa que los resultados estadísticos favorecen a ACO al encontrar el óptimo de la función en todas las ejecuciones tal como lo comprueba la desviación estándar de las 30 ejecuciones independientes. Sin embargo, ABC tiene resultados muy cercanos a la solución óptima del problema que en este caso es 0. Al realizar la prueba de Wilcoxon *Signed Rank Test* a los resultados de ambos algoritmos (mejor solución encontrada en cada una de las 30 ejecuciones independientes), se obtiene como resultado *p-value* = 0, por lo tanto, existe una diferencia significativa de los resultados entre ACO y ABC.

Para el caso de diez dimensiones, los resultados también favorecen estadísticamente al algoritmo ACO al obtener un mejor resultado que ABC y una desviación estándar mucho menor en comparación que ABC. Al realizar la prueba de Wilcoxon *Signed Rank Test* a los resultados de ambos algoritmos, la prueba da como resultado un *p-value* de 0 y un *Z-value* de -4.782. Por lo tanto, podemos concluir que existe una diferencia significativa de los resultados entre ACO y ABC.

La Figura 1 presenta el comportamiento de ambos algoritmos en el problema de la esfera con dos y diez dimensiones en la ejecución número 15 de las 30 ejecuciones realizadas de manera

Tabla 6. Resultados estadísticos de ABC y ACO en el problema de la esfera con 2 y 10 dimensiones en 30 ejecuciones independientes

	2 dimensiones		10 dimensiones	
Estadística	**ABC**	**ACO**	**ABC**	**ACO**
Mejor	9.34E-20	0.00E+00	7.41E-17	2.30E-80
Media	3.39E-18	0.00E+00	1.53E-16	2.19E-76
Desv. estándar	3.38E-18	0	6.30E-17	4.7E-76
Peor	1.52E-17	0.00E+00	2.96E-16	1.99E-75

Tabla 7. Valores de ABC y ACO en el problema de la esfera con 2 y 10 dimensiones en la ejecución independiente 15

	2 dimensiones		10 dimensiones	
Variables	**ACO**	**ABC**	**ACO**	**ABC**
x_1	-9.97E-164	-1.02E-09	3.32E-40	-2.29E-09
x_2	-1.12E-162	1.19E-09	1.12E-40	3.09E-09
x_3	-	-	2.86E-40	-8.74E-10
x_4	-	-	1.20E-39	1.66E-09
x_5	-	-	-9.42E-40	-1.55E-09
x_6	-	-	9.16E-40	9.73E-11
x_7	-	-	4.20E-40	-3.00E-10
x_8	-	-	-9.68E-40	-1.81E-09
x_9	-	-	1.68E-40	-5.11E-09
x_{10}	-	-	7.71E-40	-7.56E-09
$f(\vec{x})$	0	2.46E-18	5.10E-78	1.07E-16

independiente (mediana). Para el caso de dos y diez dimensiones, se puede observar que el algoritmo ACO tiene una convergencia hacia el óptimo después de pasar la mitad de las iteraciones. Por otro lado, ABC tiene una convergencia prematura en alguno de los óptimos locales del problema desde el principio de las generaciones. Sin duda, el algoritmo ABC tiene un verdadero problema de convergencia prematura.

Los valores finales de las variables y función objetivo de ambos algoritmos de las gráficas de convergencia se presentan en la Tabla 7.

De acuerdo con el análisis de complejidad de ACO (Gutjahr, 2008) y de ABC (Ji et al., 2015), el coste computacional del primero es menor que el segundo, ya que el orden de complejidad de ABC es cuadrático. Comparando los resultados obtenidos por ACO y ABC con los obtenidos por otras metaheurísticas, por ejemplo el algoritmo de Cúmulo de Partículas (PSO, por las siglas en inglés de *Particle Swarm Optimization*), se tienen los siguientes resultados de un algoritmo de PSO modificado aplicado al problema de la esfera con diez dimensiones (Wei et al., 2015): 1.11E-132 como mejor solución, 1.96E-121 como media, desviación estándar de 2.38E-122 y 6.44E-122 como peor solución.

De acuerdo con estos resultados, ACO y ABC se encuentran por detrás de PSO y otros algoritmos encontrados en el estado del arte. Sin embargo, si se aplica la prueba de Wilcoxon *Signed Rank Test* no hay diferencia significativa porque esta prueba considera irrelevantes valores con decimales muy cercanos a cero.

5. Conclusión

En este trabajo, se abordaron dos metaheurísticas de Inteligencia Colectiva (IC) para observar su comportamiento ante un problema de optimización conocido en el estado del arte como problema de la esfera y conocer los aspectos comunes y particulares que favorecen la convergencia prematura en el Algoritmo de Optimización de Colonia de hormigas (ACO) y Colonia Artificial de Abejas (ABC). Ambas metaheurísticas fueron programadas y adaptadas para resolver el problema de la esfera con dos y diez dimensiones usando el lenguaje de programación M empleando el software libre Octave 4.2.1. Un conjunto de experimentos fue realizado donde ambas metaheurísticas se ejecutaron con igual número de población e iteraciones para llevar a cabo una comparación justa usando medidas de rendimiento estadísticas y la prueba no paramétrica Wilcnxon *Signed Rank Test*, así como gráficos de convergencia.

Los primeros cuatro experimentos consistieron en probar a ACO y ABC en el problema de la esfera con dos y diez dimensiones con distinta población e iteraciones. Los mejores resultados obtenidos en estos experimentos fueron obtenidos con una población de 20 y 1000 iteraciones para ACO; para ABC la mejor combinación de parámetros es una población de 50 y 100 iteraciones, tomando como referencia al problema con una dimensionalidad mayor, es decir, el problema de la esfera con diez dimensiones. Un aspecto común encontrado fue que para ambas metaheurísticas resolver el problema de la esfera con diez dimensiones se vuelve más complejo que con sólo dos dimensiones.

Se observó que ACO obtiene mejores resultados cuando tiene un mayor número de iteraciones y menor número de población, es decir, entre menos hormigas y más tiempo, se acarrea más comida. En el caso de ABC, ocurre justo lo contrario: entre menos tiempo y más abejas, se recolecta más miel. Por lo tanto, una convergencia prematura ocurrirá a ambos algoritmos cuando la dimensionalidad aumente. Otro factor que provoca la convergencia prematura para el caso de ABC es una población pequeña de abejas sin importar si se tiene un gran número de iteraciones; en el caso de ACO una convergencia prematura puede ocurrir si se tiene un número menor de iteraciones sin importar el tamaño de la población.

Ambas metaheurísticas fueron probadas con el mismo número de población e iteraciones (20, 1000 respectivamente) en el problema de la esfera con dos y diez dimensiones, donde ACO obtuvo los mejores resultados y una convergencia estable hacia el óptimo global, caso contrario a ABC donde su convergencia fue prematura al caer en óptimos locales sin ninguna mejora al final de las iteraciones. Por último, la prueba de Wilcoxon *Signed Rank Test* arrojó una diferencia significativa entre los resultados de ACO y ABC.

Como trabajo futuro, ambas metaheurísticas deben ser probadas en más problemas de optimización numérica con restricciones, además de hacer experimentos donde otros parámetros de ambas metaheurísticas, aparte de la población e iteraciones, sean calibrados para conocer si estos parámetros también impactan en la convergencia y rendimiento de la metaheurística.

AGRADECIMIENTOS

Los autores agradecemos al CONACYT (Consejo Nacional de Ciencia y Tecnología de México) el apoyo otorgado al Doctorado

Interinstitucional en Ciencias de la Computación de la Universidad Juárez Autónoma de Tabasco y la Universidad Veracruzana.

REFERENCIAS

Abbass, H. A. (2001). Mbo: marriage in honey bees optimization-a haplometrosis polygynous swarming approach. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 1, pages 207–214 vol. 1.

Arito, F. L. A. (2010). Algoritmos de optimización basados en colonias de hormigas aplicados al problema de asignación cuadrática y otros problemas relacionados. Master's thesis, Universidad Nacional de San Luis Facultad de Ciencias Físico-Matemáticas y Naturales Departamento de Informática, San Luis, Argentina.

Baykasoglu, A., Ozbakir, L., and Tapkan, P. (2007). *Artificial bee colony algorithm and its application to generalized assignment problem Swarm Intelligence*, chapter Focus on Ant and Particle Swarm Optimization. Felix T.S. Chan and Manoj Kumar Tiwari. Itech Education and Pub., Vienna, Austria.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions of Systems, Man and Cybernetics-Part B*, 26(1):29–41.

Dorzán, M., Gagliardi, E., Leguizamón, M., and Hernández Peñalver, G. (2012). Approximations on minimum weight triangulations and minimum weight pseudo-triangulations using ant colony optimization metaheuristic. *Fundamenta Informaticae*, 119(1):1–27.

Eaton, J. W. (2016). GNU Octave. Technical report, Free Software Foundation. Accessed 13-11-2016.

Eiben, A. and Smith, J. E., editors (2003). *Introduction to Evolutionary Computing*. Natural Computing Series. Springer-Verlag.

Engelbrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons.

Gutjahr, W. J. (2008). First steps to the runtime complexity analysis of ant colony optimization. *Computers & Operations Research*, 35(9):2711 – 2727. Part Special Issue: Bio-inspired Methods in Combinatorial Optimization.

Haddad, O. B. and Mariño, M. A. (2007). Dynamic penalty function as a strategy in solving water resources combinatorial optimization problems with honey-bee mating optimization (hbmo) algorithm. *Journal of Hydroinformatics*, 9(3):233 – 250.

Hernández-Ocaña, B., Pozos-Parra, M., Mezura-Montes, E., Portilla-Flores, E., Vega-Alvarado, E., and Calva-Yáñez, M. (2016). Two-swim operators in the modified bacterial foraging algorithm for the optimal synthesis of four-bar mechanisms. *Computational Intelligence and Neuroscience*, 2016(0):1–18.

Ji, J., Pang, W., Zheng, Y., Wang, Z., and Ma, Z. (2015). A novel artificial bee colony based clustering algorithm for categorical data. *PloS One*, 10(5). e0127125.

Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471.

Kennedy, J. and Eberhart, R. (2001). *Swarm Intelligence*. Morgan Kaufmann, UK.

Leguizamón, G. and Coello-Coello, C. A. (2009). Boundary search for constrained numerical optimization problems with an algorithm inspired by the ant colony metaphor. *IEEE Transactions on Evolutionary Computation*, 13(2):350–368.

Matijaš, V. D., Molnar, G., Čupić, M., Jakobovic, D., and Bašić, B. D. (2010). University course timetabling using ACO: A case study on laboratory exercises. In *Knowledge-Based and Intelligent Information and Engineering Systems - 14th International Conference, KES 2010, Cardiff, UK, September 8-10, 2010, Proceedings, Part I*, pages 100–110.

Mezura-Montes, E. and Cetina-Domínguez, O. (2012). Empirical analysis of a modified artificial bee colony for constrained numerical optimization. *Applied Mathematics and Computation*, 218(22):10943 – 10973.

Mezura-Montes, E., Cetina-Domínguez, O., and Hernández-Ocaña, B. (2010). *Mecatrónica*, chapter Nuevas Heurísticas Inspiradas en la Naturaleza para Optimización Numérica. in Ramón Silva-Espinoza, Edgar Alfredo Portilla-Flores and María Aurora Molina-Vilchis (Editors) Mecatrónica. Editorial IPN, México.

Mezura-Montes, E. and Coello-Coello, C. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194.

Vega-Alvarado, E., Portilla-Flores, E., noz Hernández, G. M., Mezura-Montes, E., Sepúlveda-Cervantes, G., and Bautista-Camino, P. (2018). A memetic algorithm based on Artificial Bee Colony for optimal synthesis of mechanisms. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 34(1):1–18.

Wei, X., Zhang, J., Zhou, D., and Zhang, Q. (2015). Multi-swarm particle swarm optimization with transfer of the best particle. *Computational Intelligence and Neuroscience*, 2015. 904713.

BIOGRAFÍAS

Betania Hernández-Ocaña, obtuvo el grado de doctor en Ciencias de la Computación en la Universidad Juárez Autónoma de Tabasco en 2016 de la cual es profesora-investigadora. Actualmente es miembro del Sistema Nacional de Investigadores Nivel I y tiene el perfil deseable PRODEP. Ha participado como responsable en proyectos de investigación con y sin financiamiento. Participa activamente en la revisión de artículos de diversas revistas indizadas, asimismo en la revisión de tesis en nivel licenciatura y posgrado. Sus intereses científicos incluyen algoritmos de inteligencia colectiva y evolutivos, optimización global y aprendizaje automático.

José Hernández-Torruco, Doctor en Ciencias de la Computación por la Universidad Juárez Autónoma de Tabasco. Docente de la División Académica de Ciencias y Tecnologías de la Información (DACyTI) de la UJAT desde el año 2000. Su área de interés es el aprendizaje automático y la minería de datos, especialmente su aplicación en la medicina. Ha publicado varios artículos en inglés y español en revistas arbitradas e indexadas, así como en congresos nacionales e internacionales. Ha sido responsable de varios proyectos institucionales y ha participado como colaborador en proyectos financiados. Ha dirigido varias tesis a nivel Licenciatura, Maestría y Doctorado.

Oscar Chávez-Bosquez, Doctor en Ciencias de la Computación y actualmente miembro del Sistema Nacional de Investigadores Nivel I. Su interés es resolver problemas de la vida real empleando algoritmos inteligentes y apredizaje profundo. Cuenta con una serie de certificaciones nacionales e internacionales en desarrollo de software y ha dirigido diversas tesis a nivel licenciatura, maestría y doctorado. Ha escrito diversos artículos en revistas de alto impacto y ha desarrollado software promoviendo siempre el uso y difusión del Software Libre.

Luis G. Montané-Jiménez, Licenciado en Informática por la Universidad Veracruzana, Maestro en Computación Aplicada por el Laboratorio Nacional de Informática Avanzada y Doctor en Ciencias de la Computación por la Universidad Veracruzana. Actualmente se desempeña como profesor de Tiempo Completo en la Facultad de Estadística e Informática de la Universidad Veracruzana (México). Sus áreas de interés son el Computer-Supported Cooperative Work (CSCW), Interacción Humano-Computadora (IHC), Sistemas Conscientes del Contexto, Visualización de Información y Desarrollo de Videojuegos.