Herramienta de optimización para la enseñanza de la heurística -ABC- en curso de pregrado

Jessica Escamilla, Gina Barajas

***Resumen*— Los algoritmos bio-inspirados han exhibido desempeños aceptables en la solución de problemas de optimización industriales y hasta hace algunos años era tema de estudio exclusivo de investigaciones y cursos de pos-grado. Actualmente, estos conceptos se han difundido desde los cursos de posgrado hacia cursos de pregrado, generando nuevos desafíos en el proceso de enseñanza-aprendizaje. Este artículo aborda el problema de la creación de sistemas basados en computador que permitan entrenar a estudiantes en diseño y parametrización de heurísticas; a fin de delimitar el alcance, se toma como referencia la heurística ABC (*Artificial Bee Colony*). La motivación para abordar este sistema, surge a partir de las dificultades reportadas dentro del programa de Ingeniería Industrial al momento de asimilar técnicas de optimización basadas en algoritmos bio-inspirados. El sistema construido cuenta con tres funcionalidades básicas: primero configurar parámetros del algoritmo y seleccionar función costo, segundo visualizar abejas durante el proceso de búsqueda de solución, y tercero generar reportes asociados al proceso de búsqueda. El desarrollo involucra cuatro etapas las cuales son: levantamiento de requerimientos, diseño de algoritmos y estructuras de datos asociadas, codificación de algoritmos y validación de desempeño de la herramienta mediante pruebas de funcionalidad. Los resultados obtenidos permiten encontrar las combinaciones de parámetros asociadas con la mejor calidad y el menor tiempo de respuesta y así validar la funcionalidad del sistema.**

I. INTRODUCCIÓN

Actualmente, la optimización abarca diversos campos de aplicación, todos ellos con un propósito similar: encontrar la solución óptima. En el alcance de este propósito surgen los algoritmos como herramienta computacional para orientar la búsqueda de la solución óptima [1]. Los algoritmos bio-inspirados han exhibido desempeños aceptables en la solución de problemas de optimización industriales [2] y hasta hace algunos años eran tema de estudio exclusivo de investigaciones y cursos de pos-grado. En la actualidad surge la necesidad de incorporar las herramientas de optimización bio-inspirada dentro del conjunto de recursos con que los estudiantes logran abordar problemas de optimización. Este propósito implica la creación de Sistemas Educativos Inteligentes (IES) [3] capaces de personalizar la instrucción de heurísticas a las necesidades del alumno. Este artículo presenta un componente fundamental del sistema IES: el simulador del algoritmo bio-inspirado. Este simulador permite asistir los procesos de enseñanza en uno de los algoritmos bio-inspirados de mayor aceptación: el algoritmo ABC (*Artificial Bee Colony*). El simulador permite configurar los parámetros del algoritmo (ciclos máximos, ciclos límite y población), observar el proceso de búsqueda y determinar el desempeño de la búsqueda en términos de tiempo de cómputo y calidad de la solución. La validación de la herramienta se realizó mediante diseño experimental orientado a determinar los mejores parámetros del algoritmo durante la operación de optimización. Específicamente, se busca determinar la mejor combinación de parámetros en relación con el tiempo de cómputo y la calidad de solución. Las funciones costo utilizadas fueron seleccionadas a partir del grupo presentado en [4]. El principal criterio de selección fue la definición de dominios finitos. Los resultados obtenidos permiten verificar la funcionalidad del sistema y su utilidad al momento de analizar efectos de parámetros sobre desempeños de búsqueda de soluciones. Se espera que este simulador sea utilizado como componente fundamental de un sistema educativo inteligente orientado a potenciar las habilidades de diseño de algoritmos bio-inspirados en aplicaciones de optimización.

II. MARCO TEORICO

El algoritmo ABC fue propuesto por Dervis Karaboga en 2005 [5]. El ABC como herramienta de optimización, proporciona un procedimiento de búsqueda en el que las coordenadas en un plano constituyen a las posiciones preferidas de alimentos que representan posibles soluciones al problema. Estas posiciones o fuentes de alimento son representadas por vectores cuya dimensión está definida por el dominio de la función costo. Las posiciones son seleccionadas y evaluadas iterativamente por las abejas artificiales y el objetivo de la abeja es descubrir las mejores posiciones o la solución óptima [6].

El tamaño de la población es establecido por el parámetro SN y existen tres estados de abejas: empleadas, observadoras y exploradoras. Las abejas en estado empleadas y observadoras eligen fuentes de alimento dependiendo de la experiencia de ellas mismas y de sus compañeras de nido. Las abejas en estado exploradoras eligen las fuentes al azar sin usar la experiencia; este es un estado transitorio de las abejas empleadas [7]. El proceso de búsqueda se divide en 2 etapas: en la primera etapa llamada inicialización, las abejas empleadas seleccionan fuentes de alimento de forma iterativa y utilizando un componente aleatorio, como se presenta en (1):

Jessica Escamilla y Gina Barajas pertenecen al programa de Ingeniería Industrial de la Universidad de La Salle - Sede Bogotá y realizaron el proyecto dentro del semillero SIGMAP- Semillero de investigación en Gestión, Mejora y Análisis de Procesos. (Email: jescamilla02@unisalle.edu.co, gbarajas24@unisalle.edu.co).

El proyecto fue asesorado por Guillermo Camacho (Email: gacamacho@unisalle.edu.co)

$$x_{mi} = l_i + rand(0,1) * (u_i - l_i), \quad \textbf{(1)}$$

dónde x_{mi}, son todos los vectores de la población de fuentes de alimentos, siendo m el tamaño de la población e i las dimensiones del espacio de búsqueda, la función rand (0,1) genera un número real, aleatorio y con rango [0 - 1], l_i y u_i son el límite inferior y superior del parámetro x_{mi}, respectivamente.

En la segunda etapa, las abejas observadoras ponderan las fuentes seleccionadas por las empleadas a partir de la calidad de su néctar. Para ello, se calcula una probabilidad para cada una de las fuentes seleccionadas usando (2) y (3).

$$Fit_m(\overrightarrow{x_{mi}}) = \begin{cases} \frac{1}{1+F_{mi}(\overrightarrow{x_{mi}})} & \text{si } F_{mi}(\overrightarrow{x_{mi}}) \geq 0 \\ 1 + abs\left(F_{mi}(\overrightarrow{x_{mi}})\right) & \text{si } F_{mi}(\overrightarrow{x_{mi}}) < 0 \end{cases} \quad \textbf{(2)}$$

$$P_m = \frac{Fit_m(\overrightarrow{x_m})}{\sum_{m=1}^{SN} Fit_m(\overrightarrow{x_m})}, \quad \textbf{(3)}$$

dónde $F_m(x_m)$ es el valor de la función objetivo al considerar la solución x_m. Posteriormente, se calcula una nueva fuente de alimento v_{mi} usando (4):

$$v_{mi} = x_{mi} + \phi_{mi}(x_{mi} - x_{ki}), \quad \textbf{(4)}$$

dónde x_k es la coordenada de la posición de la fuente de alimento seleccionada inicialmente, k es un índice de parámetros elegido de manera aleatoria con distribución probabilística uniforme y ϕ_{mi} es un número aleatorio dentro del rango $[-1,1]$. Finalmente, si el valor de la calidad de esta nueva posición mejora la anterior, entonces la posición de la abeja se actualiza, de lo contrario, las abejas exploradoras incrementan el número de ensayos de esa fuente de alimento usando un contador, si el valor en este contador es mayor que un umbral $Clim$, esta fuente de alimento x_i se descarta y una abeja exploradora selecciona al azar una nueva fuente de alimento.

III. METODOLOGÍA

La metodología utilizada se compuso de 3 fases: (1) recolección de requerimientos, (2) documentación de implementaciones de ABC y (3) diseño y codificación de la solución.

A. *Recolección de requerimientos*

El resultado de la primera fase se presenta en el diagrama de casos de uso de la

. La recolección de los requerimientos se basó en las respuestas obtenidas mediante entrevistas diseñadas previamente y realizadas hacia un docente y dos estudiantes que formaron parte un grupo de pregrado en el cual se manejó la meta heurística ABC. Los resultados obtenidos permiten concluir un grupo de 3 características funcionales para el simulador: (1) configuración de parámetros que permite al usuario la posibilidad de establecer los parámetros incluyendo la posibilidad de fijar un rango de valores con un paso entre valores o un único valor, (2) opción de implementar el uso del graficador ya que la intervención de este puede afectar el tiempo de cómputo registrado, (3) selección de la función costo a optimizar.

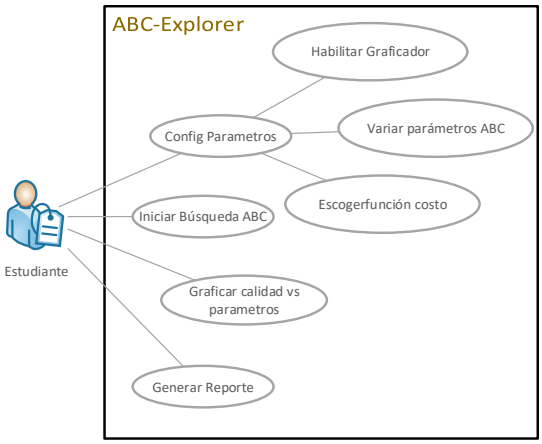

Figura 1. Diagrama de Casos de Uso.

Las características listadas, son utilizadas como insumo para el diseño del diagrama de casos de uso presentado en Figura 1. El caso de uso “iniciar búsqueda” ABC permite al usuario iniciar la búsqueda de la solución al problema de optimización. El caso de uso “graficar calidad vs parámetros” permite presentar reportes gráficos de la relación entre las salidas obtenidas y los parámetros configurados. El caso de uso “generar reporte” permite al usuario tener un reporte textual de los resultados obtenidos, específicamente: las coordenadas de la solución óptima halladas por el algoritmo, comparadas con las coordenadas de la función reportadas en la literatura, la calidad de los resultados, y el tiempo de cómputo de la solución.

B. *Documentación de implementaciones de ABC*

Durante la segunda fase se abordó la búsqueda de información necesaria para la implementación del algoritmo ABC, la información sobre los tipos de graficadores y selección del más adecuado, por último, la selección del entorno de desarrollo. Se destacan los siguientes trabajos.

En [8] presentan los resultados de comparación sobre el rendimiento del algoritmo ABC para problemas de optimización restringida aplicados a un conjunto de problemas limitados. Concluyen que el algoritmo ABC se puede usar de manera eficiente para resolver problemas de optimización restringidos

En [9] proponen un algoritmo ABC modificado (denotado como ABC / mejor), que se basa en que cada abeja busca solo alrededor de la mejor solución de la iteración anterior para mejorar la explotación. Además, para mejorar la convergencia global, cuando se produce la población inicial y se exploran las abejas, se emplean tanto los sistemas caóticos como el método de aprendizaje basado en la oposición.

En [10] proponen acelerar la ejecución de aplicaciones que requieren una gran cantidad de procesamiento, específicamente problemas matemáticos de gran complejidad. En particular, se implementó y evalúo un algoritmo ABC para resolver el problema del Set Covering, problema matemático bien definido y también un problema interesante en la teoría de la complejidad computacional cuyo estudio ha llevado al desarrollo de técnicas fundamentales para todo el campo de los algoritmos de aproximación[11]

En [12] proponen la implementación del algoritmo ABC para buscar las mejores ganancias de un controlador PID (Proporcional Integral Derivativo) Bilineal para actuadores SMA (*Shape memory alloy*), es decir, materiales metálicos que demuestran la capacidad de volver a una forma o tamaño previamente definido cuando se someten al procedimiento térmico apropiado [13]

Finalmente, destacamos el trabajo en [14] el cual, tiene como principal objetivo mostrar cómo aplicar el algoritmo de Colonia Artificial de Abejas ABC, para solucionar problemas de la vida real. Mas específicamente como es utilizada para la planificación de red en WIMAX (*Worldwide interoperabilly for Microwave Acces*),

C. Diseño y codificación de la solución

En la tercera fase se diseña y codifica el prototipo, es decir, el software necesario en el lenguaje C++, bajo el paradigma de orientación a objetos. Las principales clases diseñadas se presentan desde la Figura 3 hasta la **¡Error! No se encuentra el origen de la referencia.**.

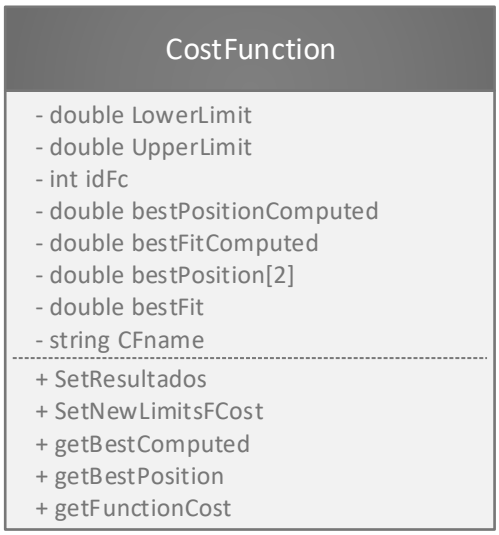

Figura 3. Diagrama de clases función costo.

Algoritmo principal

```
int main ()
{
   par=loadPar ();
   costFunction FCost(par);
   t_ini = clock ();
   faseInicializacion (Fuentes, FCost, par.SN);
   do{
      faseEmpleadas (Fuentes, FCost,par.SN);
      faseExploradoras(Fuentes, FCost,par.SN);
      faseScouts (Fuentes, FCost,par);
      ActualizaSalidas (FCost, Fuentes, par);
      DibujaFAlimentos (Fuentes, FAnt, par.SN, FCost);
      updFant (Fuentes, FAnt, par.SN);
      numeroIteraciones++;
      }
   while (numeroIteraciones<par.Cmax);
   t_fin = clock ();
   return 0;
 }
```

Figura 2. Pseudocódigo implementado utilizando c++

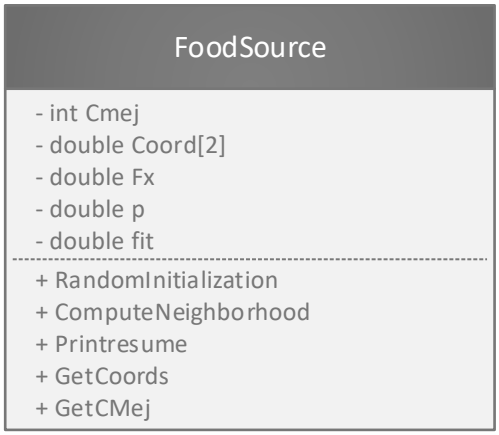

Figura 4. Diagrama de clases de las fuentes de alimentos.

Figura 5: Diagrama de la estructura parámetros.

El algoritmo utiliza la clase "función costo", la cual, representa las propiedades asociadas con: los límites superiores e inferiores, la posición para la mejor solución, el identificador de función costo y la calidad y mejor solución encontrada para cada iteración. Adicionalmente, contiene métodos para: acceso, mutación, cálculo de la mejor solución, cálculo de calidad de solución y conteo de los intentos de mejora de una solución.

En la clase $FoodSource$ están los descriptores de las fuentes de alimentos: dominio entre los límites de la función costo escogida, coordenadas óptimas de solución, la función costo evaluada en las coordenadas, la calidad de cada fuente de alimento, la probabilidad evaluada y la mejor solución

encontrada por el programa. Además, se cuenta con algunos métodos definidos para: inicializar las fuentes de alimentos principales y las fuentes de alimentos vecinas, validar mejora en cada iteración y en caso de no hallarla, abandonar la fuente de alimento e imprimir los resultados obtenidos.

En la estructura Parámetros ABC se encuentran todos los parámetros ingresados por el usuario: la cantidad de abejas que se usarán en el programa (SN), la cantidad de iteraciones que realizará el programa hallando mejores soluciones posibles ($Cmax$), la cantidad de ciclos que tiene el programa para mejorar una solución encontrada ($Clim$) y el identificador de la función costo que seleccionado. Las características de la función establecida, principalmente su respectivo dominio y valor óptimo, ya se encuentran preestablecidos dentro de la codificación del programa ($idFC$).

El pseudocódigo correspondiente se encuentra en la **¡Error! No se encuentra el origen de la referencia.**, donde se observan las principales variables y fases implementadas: inicialización, empleadas, exploradoras y Scouts. También se observan otras funciones usadas para satisfacer el requerimiento de búsqueda con el algoritmo ABC como lo son: actualizar salidas, dibujar fuentes de alimento y registro del tiempo empleado para cada iteración.

IV. RESULTADOS

En esta sección se presentan los resultados de validación de la herramienta construida. La presentación se compone de tres partes: selección de funciones costo, diseño experimental y datos registrados durante el experimento.

A. *Selección de funciones costo*

Para validar la funcionalidad de la herramienta propuesta, se seleccionaron funciones costo bajo tres requerimientos: (1) valores óptimos conocidos, (2) dominios finitos, (3) funciones de uso común en la literatura especializada. Como se puede observar en el CUADRO 1, se encontraron trece funciones costo que satisfacen estos requerimientos.

CUADRO 1. FUNCIONES REPORTADAS POR LA LITERATURA CON DOMINIO FINITO [4].

Función Costo	Mínimo Global	Dominio de Búsqueda
Goldstein–Price	f(0,-1)=3	$-2 \leq x,y \leq 2$
Sphere	f(0,0)=0;	$-5 \leq x,y \leq 5$
Eggholder	f(512,404.2319)= -959.6407;	$-512 \leq x,y \leq 512$
Damavandi	f(2,2)=0;	$-14 \leq x,y \leq 14$
Ackleys	f(0,0)=0;	$-5 \leq x,y \leq 5$
Beales	f(3,0.5)=0;	$-4.5 \leq x,y \leq 4.5$
Booth's	f(1,3)=0;	$-10 \leq x,y \leq 10$
Matyas	f(0,0)=0;	$-10 \leq x,y \leq 10$
Levi	f(1,1)=0;	$-10 \leq x,y \leq 1$
Three-hump camel	f(0,0)=0;	$-5 \leq x,y \leq 5$
Easom	f(π,π)=-1;	$-100 \leq x,y \leq 100$
Schaffer 2	f(0,0)=0;	$-100 \leq x,y \leq 100$
Schaffer 4	f(0,125313)= 0.292579;	$-100 \leq x,y \leq 100$

B. *Diseño experimental*

Inicialmente se definió el objetivo del experimento: determinar la influencia de los factores (parámetros) dentro de las variables de respuesta (calidad de solución y tiempo de cómputo). Una vez se estableció el objetivo, se fijó un rango de variabilidad para cada factor; este rango se presenta en el CUADRO 2. Finalmente se calculó el tamaño del experimento obtenido a partir de las variaciones en los factores, concluyendo 23.750 observaciones/réplicas. Este tamaño es el mismo para cada función costo. Aunque el experimento se realizó para las 13 funciones costo en CUADRO 1, por limitaciones de extensión, este artículo se concentra en el resultado obtenido con una única función costo: *Beales Function*, cuya expresión analítica se registra en (5)

$$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1{x_2}^2)^2 + (2.625 - x_1 + x_1{x_2}^3)^2 , \quad \textbf{(5)}$$

siendo x_1 y x_2 los valores posibles de las coordenadas de nuestra función

Los resultados obtenidos con las trece funciones, así como los códigos fuente, pueden ser consultados en el repositorio [15].

CUADRO 2. DIAGRAMA DEL DISEÑO DE EXPERIMENTOS REALIZADO.

Objetivo del experimento	Determinar la influencia de los factores, cantidad de abejas ($\boldsymbol{S}$), ciclos máximos de iteración ($\boldsymbol{Cmax}$) y ciclos límite de mejora ($\boldsymbol{Clim}$), en las variables de respuesta calidad y tiempo del algoritmo ABC
Variable de respuesta	Calidad y tiempo de búsqueda
Factores de diseño	S que varía desde 10 hasta 100 en intervalos de 5. 19 niveles $Cmax$ que varía desde 10 hasta 500 en intervalos de 10. 50 niveles $Clim$ que varía desde 2 hasta 50 en intervalos de 2. 25 niveles
Unidad experimental	Función costo *Beale*
Estrategia de bloqueo	La función costo= Beale Function Sus límites correspondientes [-4.5 , 4.5]
Tamaño del experimento	El tamaño del experimento es de 23.750 (19*25*50) observaciones.

Para este experimento se variaron sistemáticamente los parámetros de entrada del algoritmo obteniendo así todos nuestros posibles resultados.

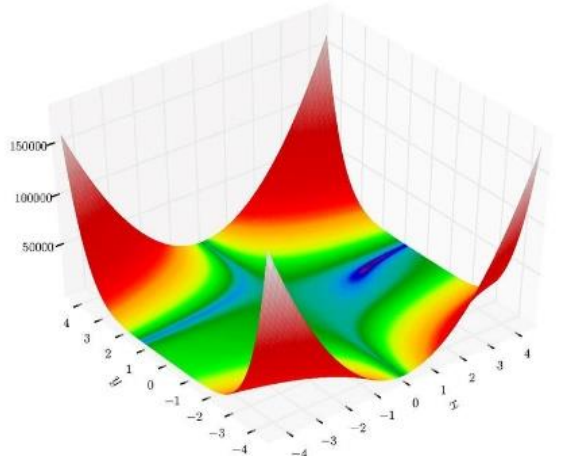

Figura 6: *Function Beale* [16]

Las variables controlables durante el experimento fueron: la memoria *RAM*, y el procesador, ya que, si se usan computadores con características diferentes, los resultados cambiarán dependiendo de dichas propiedades. Uno de los parámetros no controlados es el generador de números aleatorios existente en el algoritmo. El objetivo principal del experimento diseñado consiste en la evaluación de los resultados obtenidos dependiendo de los valores establecidos por los parámetros, por lo tanto se realizaron la diferentes combinaciones posibles teniendo en cuenta los rangos establecidos inicialmente para cada parámetro; con los resultados obtenidos se evaluó la calidad por medio de la comparación entre los resultados obtenidos por los algoritmos y los resultados del óptimo definido teóricamente por la literatura de la función costo. Adicionalmente se tomó la diferencia entre estos datos y se ubicó en una escala de 0 a 1 indicando que 0 es mala calidad (muy alejado del optimo) y 1 buena calidad (muy cerca o el valor exacto del optimo)

C. Datos registrados

Se realizaron todas las combinaciones entre los tres parámetros sin repetición, sin embargo, para la observación adecuada de la calidad de cada resultado, se requiere de un parámetro que permanezca constante. Por ende, se realizaron tres gráficas variando respectivamente: $clim$ – $cmax$ (ver Figura 7), $cmax$ – s (ver Figura 8) y s – $clim$ (ver Figura 9). En cada una de ellas el parámetro faltante es constante. El valor de este parámetro es escogido en el punto donde los resultados se empiezan a comportar de manera estable.

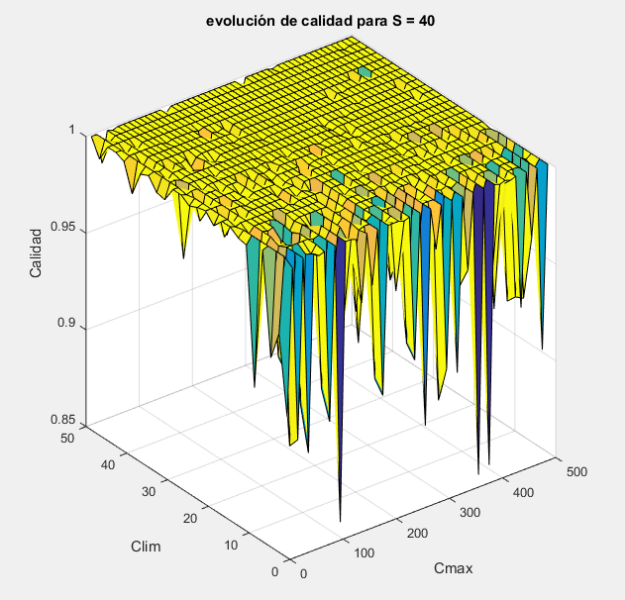

Figura 7. Resultados obtenidos variando los parámetros $Clim$ y $Cmax$ observando la calidad de los resultados en un rango de 0 a 1.

Como se puede observar en la Figura 7 al aumentar el valor del ciclo límite la calidad obtenida en los resultados mejora. De otro lado, para ciclos límites cercanos a cero, el algoritmo no logra encontrar los valores óptimos, aun cuando utilice altos valores de iteraciones ($Cmax$).

La Figura 8 presenta el efecto de variar el valor del ciclo máximo $Cmax$ y la cantidad de abejas S. La calidad de la solución es altamente sensible en la zona de bajos valores de S y $Cmax$. Por el contrario, esta calidad no presenta mayores variaciones en valores altos de S y $Cmax$. La influencia de la cantidad de abejas es mínima en comparación con los ciclos máximos, a medida que el algoritmo va iterando en ambos parámetros se observa la mejora continua en la calidad.

Como se puede observar en la Figura 9 el aumento en el valor del ciclo límite, la calidad obtenida en los resultados tiene una mejora mínima, sin embargo, la influencia de la cantidad en la población se puede observar en una gran mejora comparándolo con los ciclos límites, a medida que el programa va iterando en ambos parámetros se observa la mejora continua en la calidad.

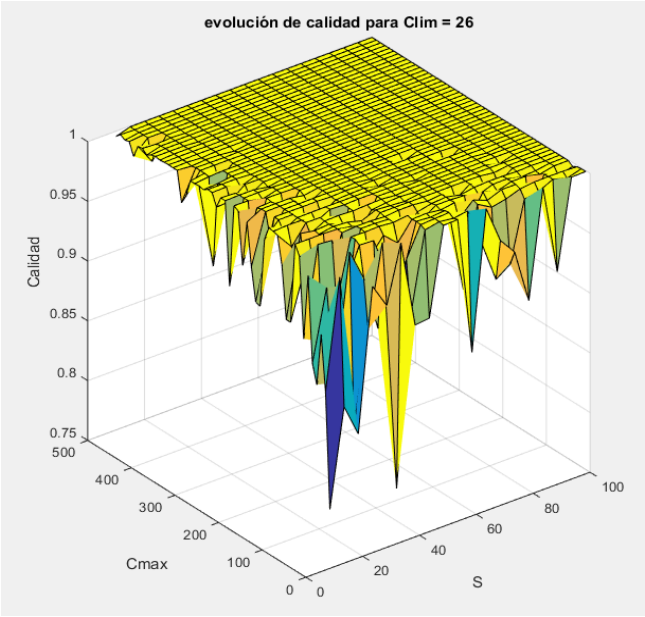

Figura 8. Resultados obtenidos variando los parámetros S y $Cmax$ observando la calidad de los resultados en un rango de 0 a 1.

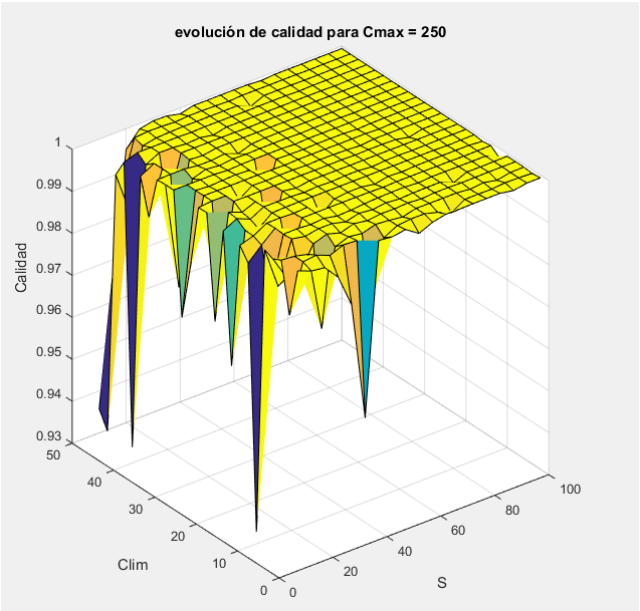

Figura 9. Resultados obtenidos variando los parámetros S y $Clim$ observando la calidad de los resultados en un rango de 0 a 1.

Comparando los resultados en las tres gráficas, entre mayor sean los valores establecidos en los parámetros, los resultados toman un comportamiento más uniforme y más cercanos al óptimo. La diferencia entre el valor óptimo real y el valor calculado por ABC tiene un 1% de error promedio.

CONCLUSIONES

En este trabajo se presentó un simulador diseñado para asistir procesos de enseñanza de la meta heurística ABC en cursos de pregrado. El proceso de diseño involucró etapas de: levantamiento de requerimientos, documentación e implementación del algoritmo ABC, diseño y codificación del simulador, y validación del simulador. La herramienta construida permite realizar un análisis (1) independiente o (2) sistemático del algoritmo ABC. En el primero, la herramienta se enfoca en la diagramación del desplazamiento de las abejas durante una tarea de optimización con parámetros fijos. En el segundo implica múltiples búsquedas de valor óptimo, cada una con diferentes configuraciones en los parámetros del algoritmo. Se espera que estas características faciliten el entendimiento del efecto de parámetros sobre el proceso de búsqueda de solución óptima con el algoritmo ABC

El proceso de validación se realizó mediante diseño experimental buscando determinar los mejores parámetros de ABC durante la optimización de una función costo de prueba. Los resultados obtenidos permiten concluir la dependencia entre calidad de solución, parámetros de configuración del algoritmo y validación de la funcionalidad de la herramienta.

Para concluir respecto de la utilidad de la herramienta en procesos de enseñanza, se requiere la prueba dentro de los espacios académicos. Por medio de la experimentación se identifican limitaciones del simulador entre las que desatacamos: interfaz de difícil usabilidad, imposibilidad de adicionar funciones costo de usuario, limitación de funciones costo a dos dimensiones. Se planea abordar estas limitaciones y la prueba dentro de espacios académicos, en trabajos futuros.

Como trabajos futuros se espera: (1) ajustar las limitaciones identificadas en interfaz, usabilidad, adición de funciones costo y restricción de dimensiones de búsqueda, (2) incluir el simulador presentado dentro de un SIE (Sistemas Educativos Inteligentes) que permita personalizar la instrucción de heurísticas a las necesidades del alumno, (3) realizar pruebas dentro de los espacios académicos a fin de concluir respecto de su utilidad en el proceso de enseñanza-aprendizaje, específicamente, en la habilidad de establecer relaciones entre parámetros de configuración y desempeño de la meta-heurística durante el proceso de optimización.

REFERENCIAS

[1] O. Suárez, "Una aproximación a la heuristica y metaheuristicas," *Red Colomb. Rev. Ing.*, pp. 44–51, 2011.

[2] E. Flórez, N. Díaz, W. Gómez, L. Bautista, and D. Delgado, "Evaluación de algoritmos bioinspirados para la solución del problema de planificación de trabajos1 Evaluation of bioinspired algorithms for the solution of the job scheduling problem .," *Rev. Investig. I+D*, vol. 11, no. 1, pp. 143–155, 2018.

[3] L. Aroyo, A. Graesser, and W. L. Johnson, "Guest Editors' Introduction: Intelligent Educational Systems of the Present and Future," no. August, pp. 6–8, 2007.

[4] WikiProject Mathematics, "Test functions for optimization," *Wikipedia, la enciclopedia libre*. [Online]. Available: https://en.wikipedia.org/wiki/Test_functions_for_optimization.

[5] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," 2005.

[6] M. G. Aza, "Implementación de algoritmos de Inteligencia Swarm con fines didácticos," 2013.

[7] A. Vidal, "Algoritmos Heuristicos en Optimizacion," p. 94, 2013.

[8] D. Karaboga and B. Basturk, "Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization," in *Foundations of Fuzzy Logic and Soft Computing*, Springer, Berlin, Heidelberg, 2007, pp. 789–798.

[9] G. Weifeng, S. Liu, and H. Lingling, *Journal of Computational and Applied Mathematics*. 2012.

[10] M. Herrera and M. Salas, "Implementación y evaluación de un algoritmo abc en gpu," p. 28, 2014.

[11] T. Stern, "What is the set cover problem," in *Set Cover Problem*, 2000, pp. 1–5.

[12] Z. Ye Fu, "Implementación de Artificial Bee Colony para Controlador Bilineal en actuadores SMA," Universidad Carlos III de Madrid.

[13] K. Shimizu and T. Tasaki, "Shape Memory Alloys and Wires (SMA)," 2012. [Online]. Available: https://www.reade.com/products/shape-memory-alloys-and-wires-sma.

[14] A. Araujo, A. Méndez, and T. Moreno, "Algoritmo de Enjambre de Abejas," Universidad de los Andes Mérida, 2013.

[15] J. Escamilla and G. Barajas, "Algoritmo-ABC," *GitHub*, 2018. [Online]. Available: https://github.com/JessicaEscamilla/Algoritmo-ABC.

[16] D. B. Sonja Surjanovic, "Test Functions and Data Sets," *Simon Fraser University*, 2013. [Online]. Available: https://www.sfu.ca/~ssurjano/beale.html.