

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Técnica Meta-heurística de optimización inspirada en las características y movimiento de los pulpos

EDUARDO ANDRÉS GONZÁLEZ GUZMÁN

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Mecánica y Mecatrónica
Bogotá, Colombia
2018

Técnica Meta-heurística de optimización inspirada en las características y movimiento de los pulpos

EDUARDO ANDRÉS GONZÁLEZ GUZMÁN

Tesis presentada como requisito parcial para optar al título de:
Magister en Ingeniería Mecánica

Directora:

MARÍA ALEJANDRA GUZMÁN PARDO, PhD.

Línea de Investigación:

Algoritmos Metaheurísticos de Optimización

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería Mecánica y Mecatrónica

Bogotá, Colombia

2018

“I'm not special. I've never been exceptional. This is, though. What I'm doing. My work.”

Jonathan Doe, movie Se7en.

Agradecimientos

A Dios y a todos los espíritus que me iluminaron y cuidaron de mí para poder realizar exitosamente este trabajo.

A mi Padre Ángel González, por el respaldo y apoyo incondicional brindado; ya que ha sido fundamental para alcanzar este objetivo.

A mi madre Gloria Guzmán, por la paciencia y colaboración durante este proceso.

A la Dr. María Alejandra Guzmán Pardo, por el respeto a mis sugerencias e ideas y por la dirección que ha facilitado la culminación de este trabajo.

A Lorena, por enseñarme que los compromisos adquiridos deben ser atendidos sin culpar a otros por las dificultades.

A Fabián, por su soporte en los momentos de duda.

A William, por la ayuda técnica brindada.

A Jeffrey, por los consejos de redacción.

A todos mis estudiantes, quienes involuntariamente me motivaron a continuar esta búsqueda de conocimiento, con el fin de poder brindarles una mejor enseñanza.

A Luis, Manuel, Johan, Improvement Pill, Dr. Jordan Peterson, Diego Dreyfus y todos los que de alguna manera hicieron posible la realización de este documento.

Resumen

En este documento se desarrolla un nuevo algoritmo de optimización metaheurística para problemas mono-objetivo con variables continuas, denominado algoritmo de optimización del pulpo artificial (AOOA por sus siglas en inglés); el cual emula el movimiento y características anatómicas de los pulpos.

La técnica es evaluada mediante un conjunto de 42 funciones de prueba, las respuestas encontradas muestran la habilidad del algoritmo para explorar y explotar eficientemente el espacio de búsqueda; logrando la obtención del mínimo global en la mayoría de las 42 funciones consideradas. Al comparar AOOA contra técnicas convencionales de optimización metaheurística, se encontró que el algoritmo diseñado presenta resultados competitivos con respecto a las otras técnicas analizadas; demostrando que es un instrumento válido y eficaz en el campo de la optimización metaheurística. Adicionalmente se realizó un análisis de sensibilidad de respuesta, desviación estándar y tiempo de ejecución del algoritmo, en función de la configuración de sus dos parámetros.

Este algoritmo se emplea también en la solución de un problema de diseño mecánico con restricciones, encontrando resultados acertados con una desviación estándar baja; verificando que AOOA es eficaz para la solución de problemas reales de ingeniería con restricciones, ofreciendo además robustez en sus soluciones.

Con base en los estudios y análisis realizados, se concluye que AOOA es una técnica robusta de optimización metaheurística, capacitada para resolver problemas de optimización mono-objetivo con variables continuas, con o sin restricciones; encontrando resultados adecuados con una alta tasa de éxito.

Palabras clave: Pulpos, Optimización, Metaheurísticas, Algoritmos bioinspirados, Técnicas de optimización.

Abstract

In this document is developed, a new metaheuristic optimization algorithm for mono-objective problems with continuous variables, called Artificial Octopus Optimization Algorithm (AOOA); which emulates the movement and anatomical characteristics of the octopuses.

The technique is evaluated through a set of 42 test functions, the answers found show the algorithm's ability to efficiently explore and exploit the search space; obtaining the global minimum in most of the 42 functions considered. When comparing AOOA against conventional metaheuristic optimization techniques, it was found that the designed algorithm presents competitive results with respect to the other techniques analyzed; demonstrating that it is a valid and effective instrument in the field of metaheuristic optimization. Additionally, an analysis of response sensitivity, standard deviation and execution time of the algorithm was performed, based on the configuration of its two parameters.

This algorithm is also used in the solution of a mechanical design problem with restrictions, finding successful results with a low standard deviation; verifying that AOOA is effective for the solution of real engineering problems with restrictions, offering also robustness in its solutions.

Based on the studies and analyzes carried out, it is concluded that AOOA is a robust metaheuristic optimization technique, capable of solving mono-objective optimization problems with continuous variables, with or without restrictions; finding suitable results with a high success ratio.

Keywords: Octopuses, Optimization, Metaheuristics, Bioinspired Algorithms, Optimization Techniques.

Índice General

	Pág.
Resumen	V
Abstract.....	VI
Lista de Figuras	IX
Lista de Tablas.....	XI
Lista de Abreviaturas y Símbolos.....	XIII
Introducción	XIX
Objetivos de la Tesis	XXI
Organización de la Tesis	XXII
1. Marco Teórico	23
1.1 Conceptos generales de optimización	23
1.2 Clasificación de técnicas en optimización	25
1.3 Heurística y metaheurística.....	26
1.4 Algoritmos representativos de optimización metaheurística.....	26
1.4.1 Algoritmo genético (Genetic Algorithm-GA)	26
1.4.2 Algoritmo de la colonia de hormigas (Ant Colony Optimization-ACO)	28
1.4.3 Optimización por enjambre de Partículas (Particle Swarm Optimization-PSO).....	29
1.4.4 Evolución diferencial (Differential Evolution-DE)	29
1.4.5 Algoritmo de optimización del forrajeo de bacterias (Bacterial Foraging Optimization Algorithm-BFOA)	31
1.4.6 Algoritmo de colonia artificial de abejas (Artificial Bee Colony-ABC)..	32
1.4.7 Algoritmo de luciérnagas (Firefly Algorithm-FA).....	34
1.5 Algoritmos establecidos en la última década	34
1.5.1 Algoritmo de optimización reacción química artificial (Artificial Chemical Reaction Optimization Algorithm-ACROA)	35
1.5.2 Algoritmo de optimización con base en la enseñanza y el aprendizaje (Teaching-Learning-Based Optimization-TLBO)	35
1.5.3 Algoritmo de pastoreo de Kril (Krill Herd-KH)	36
1.5.4 Algoritmo del ciclo del agua (Water Cycle Algorithm-WCA).....	36
1.5.5 Algoritmo de optimización búsqueda de pingüinos (Penguins Search Optimization Algorithm-PeSOA)	37
1.5.6 Algoritmo de optimización del arrecife de coral (The Coral Reefs Optimization-CRO)	37
1.5.7 Algoritmo búsqueda de organismos simbióticos (Symbiotic Organisms Search-SOS)	38
1.5.8 Algoritmo de optimización del alga artificial (Artificial Algae Algorithm-AAA)	39
1.5.9 Algoritmo de optimización de la ballena (The Whale Optimization Algorithm-WOA).....	40

1.5.10	Algoritmo de optimización del saltamontes (Grasshopper Optimisation Algorithm-GOA).....	40
1.6	Características principales de las metaheurísticas.....	41
2.	Metaheurística Diseñada	42
2.1	Descripción general de los pulpos	42
2.2	Características y movimientos emulados de los pulpos en la técnica metaheurística creada	43
2.2.1	Distribución espacial del animal	44
2.2.2	Habilidades cognitivas del pulpo	47
2.2.3	Dimensión longitudinal de las extremidades del animal	48
2.2.4	Movimiento de extremidades para alcanzar objetos.....	50
2.2.5	Regeneración de miembros perdidos.....	51
2.2.6	Movimientos para el desplazamiento (Locomoción)	54
2.3	Algoritmo de optimización del pulpo artificial (Artificial Octopus Optimization Algorithm-AOOA)	58
2.3.1	Pseudocódigo de AOOA.....	59
2.3.2	Diagrama de flujo de AOOA	60
3.	Desempeño del Algoritmo.....	61
3.1	Problemas de prueba utilizados para la evaluación del algoritmo.....	61
3.2	Evaluación de desempeño con respecto a problemas de prueba	63
3.3	Evaluación de desempeño con respecto a otros algoritmos.....	70
3.3.1	Prueba comparativa 1 AOOA vs GA, DE, PSO y ABC.....	70
3.3.2	Prueba comparativa 2 AOOA vs estrategias evolutivas y ABC.....	74
3.4	Selección de parámetros adecuados en AOOA y análisis de sensibilidad de respuesta.....	78
4.	Aplicación a un Problema de Ingeniería	88
4.1	Problema de optimización de la viga soldada	88
4.2	Resultados con AOOA al problema de diseño de la viga soldada	90
4.3	AOOA frente a otras técnicas de optimización para el problema de diseño de la viga soldada	91
5.	Conclusiones	93
5.1	Contribuciones	93
5.2	Resultados	94
5.3	Trabajos Futuros	95
Anexo A: Implementación del algoritmo en el lenguaje de MATLAB.....		96
Anexo B: Resultados de las 30 ejecuciones del algoritmo en el problema de aplicación del diseño de la viga soldada		107
Bibliografía.....		108

Lista de Figuras

	Pág.
Figura 1.1 Clasificación de las técnicas de optimización, modificada de [21].....	25
Figura 2.1 (a) Fotografía de la especie <i>Octopus Vulgaris</i> . (b) Fotografía Pulpo Arrastrándose. Fotos: Pixabay.....	42
Figura 2.2 Designación de los brazos del pulpo [65].....	44
Figura 2.3 Arreglo general de ventosas de una hilera. (a) Vista del arreglo de ventosas en <i>E. moschata</i> . (b) Distribución de las ventosas alrededor del pico (recuadro negro en a). (c) Subdivisión de las ventosas en un solo brazo [67].....	44
Figura 2.4 Esquema de representación del pulpo virtual en el espacio de búsqueda Ω ; caso específico para un problema de optimización con dos dimensiones y tres ventosas por brazo.....	46
Figura 2.5 Variación de la posición de la mejor ventosa por brazo (color azul); a medida que el indicador j varia. Izquierda caso para $j=1$; tentáculos 4 y 8 poseen un valor $Bsa = Bt$. Derecha caso para $j=2$; el tentáculo 8 aún posee un valor $Bsa = Bt$	48
Figura 2.6 Secuencia de video que muestra 4 imágenes de un pulpo alcanzando un pequeño disco plástico con 2 brazos [75].	50
Figura 2.7 Grafico en \mathbb{R}^2 que representa la emulación de la orientación de extremidades del pulpo; en rojo se observan los segmentos que sufrieron orientación y elongación, en azul las ventosas Bsa	51
Figura 2.8 Imágenes de la evolución de la extremidad mutilada. En el recuadro A, se observa el animal de control; en los otros recuadros se muestra la evolución del brazo mutilado desde el día 3 al 55 [81].....	52
Figura 2.9 Gráfico en \mathbb{R}^2 que representa la acción del operador regeneración 1. Se muestra únicamente uno de los brazos afectados; donde las ventosas $S11$ y $S12$ sufrieron cambio.	53
Figura 2.10 Grafico en \mathbb{R}^2 que representa la acción del operador regeneración 2 en un brazo seleccionado aleatoriamente; donde la ventosa $S13$ se cambió en su totalidad.	53

Figura 2.11 Locomoción del pulpo <i>Abdopus aculeatus</i> . (a) Arrastramiento; (b) Propulsión a chorro [82].	55
Figura 2.12 Imagen de un pulpo virtual 2D, en el cual se muestra el proceso de establecimiento de ventosas para $ns = 3$. a) Ventosas iniciales $j = 1$; b) ventosas en $j = 2$; c) ventosas para $j = ns$.	57
Figura 3.1 Curvas de convergencia de las funciones Sphere, Sum Squares, Powell y Rosenbrock. Las curvas muestran el mejor valor promedio obtenido durante cada iteración en las 30 ejecuciones del algoritmo.	66
Figura 3.2 Curvas de convergencia de las funciones Bohachevsky 1, Rastrigin, Schwefel y Michalewicz 10. Las curvas muestran el mejor valor promedio obtenido durante cada iteración en las 30 ejecuciones del algoritmo.	67
Figura 3.3 Curvas de convergencia de las funciones Six-Hump Camel, Shekel 7, Perm, y Power sum. Las curvas muestran el mejor valor promedio obtenido durante cada iteración en las 30 ejecuciones del algoritmo.	68
Figura 3.4 Curvas de convergencia de las funciones Hartman 6-D, Ackley, Langerman 5 y Fletche Powell 10. Las curvas muestran el mejor valor promedio obtenido durante cada iteración en las 30 ejecuciones del algoritmo.	69
Figura 3.5 Resultados gráficos del estudio extensivo de parámetros para las funciones SumSquares y Schwefel 2.22.	82
Figura 3.6 Resultados gráficos del estudio extensivo de parámetros para las funciones Rosenbrock y Rastrigin.	83
Figura 3.7 Resultados gráficos del estudio extensivo de parámetros para las funciones Michalewicz 10 y Shekel 7.	84
Figura 3.8 Resultados gráficos del estudio extensivo de parámetros para las funciones Power Sum y Ackley.	85
Figura 4.1 Esquema de la viga soldada. [102]	88

Lista de Tablas

	Pág.
Tabla 1. Conjunto de funciones empleadas para las pruebas experimentales. D: dimensión de la función, T: tipo de función (U: unimodal, M: multimodal, S: separable, N: no-separable), No: número.	62
Tabla 2. Resultados estadísticos de evaluar las 42 funciones de prueba con AOOA, para cien mil evaluaciones de la función objetivo. D: dimensión de la función, T: tipo de función, Min.Obt: mínimo valor obtenido, Prom.: promedio de los mejores valores obtenidos en las 30 ejecuciones del algoritmo, D. Estd: desviación estándar.	64
Tabla 3. Resultados estadísticos de las 30 evaluaciones de los algoritmos GA [91], PSO [91], DE [91], ABC [91] y AOOA. Prom.: promedio de los mejores valores obtenidos en cada evaluación, D. Estd: desviación estándar. En negrita valores mínimos.	71
Tabla 4. Resultados estadísticos de las 50 evaluaciones de los algoritmos CES [95], ESLAT [95], CMA-ES [95], ABC [91] y AOOA. Prom.: promedio de los mejores valores obtenidos, D. Estd: desviación estándar. En negrita valores mínimos.	75
Tabla 5. Costo de la solución, en función del número promedio de las evaluaciones de la función objetivo, para los algoritmos CES [95], ESLAT [95], CMA-ES [95], ABC [91] y AOOA.	76
Tabla 6. Tasa de éxito (%) de los algoritmos ESLAT [95], CMA-ES [95], ABC [91] y AOOA.	76
Tabla 7. Conjunto de funciones utilizadas para la sintonización de parámetros	79
Tabla 8. Estudio extensivo de parámetros donde cada configuración fue realizada 30 veces y promediados sus resultados.	80
Tabla 9. Mejores configuraciones de parámetros y límites de los parámetros <i>ns</i> y <i>Pr</i>	86
Tabla 10. Configuración seleccionada de parámetros para la metaheurística AOOA.	87
Tabla 11. Resultados óptimos obtenidos con AOOA para el problema de la viga soldada. .91	.91
Tabla 12. Resultados obtenidos para las restricciones en el problema de la viga soldada. .91	.91

Tabla 13 Comparación de los resultados de AOOA para el problema de optimización de la viga soldada.....	92
Tabla 14 Resultados estadísticos de diferentes técnicas de optimización, para el problema de diseño de la viga soldada.....	92

Lista de Abreviaturas y Símbolos

Abreviaturas

Abreviatura	Término
EA	Algoritmos Evolutivos (Evolutionary Algorithms-EA)
EDAs	Algoritmo de estimación de distribución (Estimation of Distribution Algorithms-EDAs)
SS	Búsqueda Dispersa (Scatter Search-SS)
SA	Recocido simulado (Simulated Annealing-SA)
VNS	Búsqueda de entorno variable (Variable Neighbourhood Search-VNS)
TS	Búsqueda Tabu (Tabu Search-TS)
GRASP	(Greedy Randomized Adaptive Search Procedure-GRASP)
ILS	Búsqueda local iterativa (The Iterated Local Search-ILS)
GA	Algoritmo genético (Genetic Algorithm-GA)
ACO	Algoritmo de la colonia de hormigas (Ant Colony Optimization-ACO)
PSO	Optimización por enjambre de Partículas (Particle Swarm Optimization-PSO)
DE	Evolución diferencial (Differential Evolution-DE)
BFOA	Algoritmo de optimización del forrajeo de bacterias (Bacterial Foraging Optimization Algorithm-BFOA)
ABC	Algoritmo de colonia artificial de abejas (Artificial Bee Colony-ABC)
FA	Algoritmo de luciérnagas (Firefly Algorithm-FA)
ACROA	Algoritmo de optimización reacción química artificial (Artificial Chemical Reaction Optimization Algorithm-ACROA)
TLBO	Algoritmo de optimización con base en la enseñanza y el aprendizaje (Teaching-Learning-Based Optimization-TLBO)
PESO	Optimización por enjambre evolutivo de partículas (Particle Evolutionary Swarm Optimization-PESO)
CDE	Evolución diferencial culturizada (Cultured Differential Evolution for constrained optimization-CDE)
KH	Algoritmo de pastoreo de Kril (Krill Herd-KH)

Abreviatura	Término
<i>WCA</i>	Algoritmo del ciclo del agua (Water Cycle Algorithm-WCA)
<i>PeSOA</i>	Algoritmo de optimización búsqueda de pingüinos (Penguins Search Optimization Algorithm-PeSOA)
<i>CRO</i>	Algoritmo de optimización del arrecife de coral (The Coral Reefs Optimization-CRO)
<i>SOS</i>	Algoritmo búsqueda de organismos simbióticos (Symbiotic Organisms Search-SOS)
<i>AAA</i>	Algoritmo de optimización del alga artificial (Artificial Algae Algorithm-AAA)
<i>ACOr</i>	Algoritmo de la colonia de hormigas para dominio continuo (Ant Colony Optimization for continuous domains-ACOr)
<i>WOA</i>	Algoritmo de optimización de la ballena (The Whale Optimization Algorithm-WOA)
<i>GOA</i>	Algoritmo de optimización del saltamontes (Grasshopper Optimisation Algorithm-GOA)
<i>AOOA</i>	Algoritmo de optimización del pulpo artificial (Artificial Octopus Optimization Algorithm-AOOA)
<i>CS</i>	Algoritmo metaheurístico búsqueda del cuco (Cuckoo Search)
<i>CES</i>	Estrategias de evolución canónicas (Canonical Evolution Strategies)
<i>ESLAT</i>	Estrategias de evolución aprendidas con terminación automática (Evolution Strategies Learned with Automatic Termination)
<i>CMA-ES</i>	Estrategias de evolución con adaptación de la matriz de covarianza (Covariance Matrix Adaptation Evolution Strategies)
<i>GSA</i>	Algoritmo de búsqueda gravitacional (Gravitational Search Algorithm)
<i>CBO</i>	Algoritmo de optimización de cuerpos colisionantes (Colliding Bodies Optimization)
<i>RO</i>	Algoritmo de optimización de rayos (Ray Optimization)
<i>IHS</i>	Algoritmo búsqueda de armonía mejorada (Improved Harmony Search)
<i>RANDOM</i>	Método aleatorio de Richardson (Richardson's Random method)
<i>SIMPLEX</i>	Método Simplex con una función de penalización
<i>DAVID</i>	Davidon-Fletcher-Powell con una función de penalización
<i>APPROX</i>	Aproximación lineal sucesiva de Griffith y Stewart's

Símbolos

Símbolo	Término
$f(\hat{\mathbf{x}})$	Función Objetivo
i, j, k	Índices utilizados para referenciar elementos de una matriz
$g_i(\hat{\mathbf{x}}), h_j(\hat{\mathbf{x}})$	Funciones para la definición de restricciones
n	Dimensiones del espacio de búsqueda Ω
m, p	Valores numéricos que pueden tomar i, j en las ecuaciones (1.1) y (1.2)
Ω	Espacio de búsqueda
$\hat{\mathbf{x}}$	Vector n-dimensional
x, x_1, \dots, x_n	Variables de decisión
\mathbb{R}^n	Espacio vectorial \mathbb{R}^n
K	Hormigas que iterativamente pasan de un estado a otro en ACO
y, z	Estados posibles de las hormigas en ACO
P_{yz}^k	Probabilidad de cambio de un estado y a z en ACO
η_{yz}	Valor que determina que tan atractivo es un movimiento en ACO
τ_{yz}	Valor que determina que tan conveniente es un movimiento en ACO
α	Parámetro que controla la influencia de τ_{yz} en ACO
β	Parámetro que controla la influencia de η_{yz} en ACO
t	Contador de cada iteración.
w	Inercia en PSO.
r_1, r_2	Términos aleatorios en PSO
c_1, c_2	Valores de aceleración en PSO
$\hat{\mathbf{b}}$	Mejor posición propia de la partícula en PSO
$\hat{\mathbf{g}}$	Mejor posición del enjambre en PSO
$\hat{\mathbf{v}}$	Velocidad de la partícula en PSO
$\hat{\mathbf{y}}$	Posición de la partícula en algún instante en PSO
NP	Población de vectores en DE
$\hat{\mathbf{m}}$	Vector perturbado en DE
D	Dimensión de los problemas de optimización
b_1, b_2, b_3, b_4	Índices aleatorios en DE, ABC y AOOA
F	Factor de ampliación en DE
$\hat{\mathbf{t}}$	Vector objetivo en DE

Símbolo	Término
\hat{e}	Vector de prueba en DE
CR	Constante de cruce en DE
d	Número aleatorio perteneciente al intervalo [0,1]
J^*	Superficie de nutrientes del ambiente en BFOA
θ	Símbolo para indicar a una bacteria en BFOA
k^*	Índice de reproducción en BFOA
l^*	Eventos de dispersión y eliminación en BFOA
P'	Posición de cada bacteria en BFOA
j^*	Índice del paso quimiotáxico en BFOA
T^*	Población en BFOA
N_c	Cantidad de pasos quimiotáxicos en BFOA
C	Paso en BFOA
Φ	Dirección aleatoria de Giro en BFOA
N_s	Número de pasos máximos antes de un giro en BFOA
P_{ed}	Probabilidad de eliminación y dispersión en BFOA
BN	Abejas Empleadas en ABC
SN	Fuentes de comida en ABC
fit	Valor de adecuación de las abejas en ABC
P^*	Probabilidad de decisión para abejas observadoras en ABC
φ	Factor en ABC perteneciente al intervalo [-1,1]
X	Posición de una luciérnaga cualquiera en FA
ϵ	Vector para FA, dibujado desde una distribución de Gauss
α	Parámetro que controla el tamaño de paso en FA
β^*	Atracción de dos luciérnagas que están juntas en FA
ε	Valor para determinar la atracción entre dos luciérnagas en FA
q	Distancia entre luciérnagas en FA
N^*	Cantidad de reactivos en ACROA
A	Matriz que contiene las especies de coral virtuales en CRO
M^*, N	Dimensiones de la matriz A en CRO
\hat{S}	Representación de una ventosa del pulpo en AOOA
\hat{B}	Representación del pico del pulpo en AOOA

Símbolo	Término
ns	Numero de ventosas en cada brazo del octópodo en AOOA
\hat{T}	Secciones de brazo del octópodo en AOOA
N_{pop}	Población total de la metaheurística AOOA
\hat{Bp}	Mejor posición del pulpo en AOOA
It_{max}	Máximo número de iteraciones en AOOA
\hat{Bs}	Mejor posición de las ventosas del pulpo en la iteración actual en AOOA
\hat{Bsa}	Mejor ventosa de cada brazo del cefalópodo en la iteración actual en AOOA
ef	Factor de alargamiento de cada segmento de brazo en AOOA
sf	Factor de encogimiento de tentáculos en AOOA
\hat{L}	Paso entre ventosas en AOOA
\hat{O}	Orientación de la secciones de brazo en el espacio en AOOA
Pr	Parámetro que controla la relación de exploración y explotación en AOOA
$rand$	Número aleatorio perteneciente al intervalo (0,1)
\hat{Y}	Arreglo de valores aleatorios en el intervalo (0 y 1)
$\hat{L}i$	Límites inferiores de las variables del problema
$\hat{L}s$	Límites superiores de las variables del problema
$\hat{\zeta}$	Vector de valores aleatorios entre (0 y 1)
$[x]$	Obtiene el entero más cercano a x (función redondeo)
\mathbb{Z}^+	Enteros Positivo
τ	Esfuerzo cortante
τ_{max}	Esfuerzo cortante máximo permitido
τ'	Esfuerzo cortante primario
τ''	Esfuerzo cortante secundario
σ	Esfuerzo de flexión
σ_{max}	Esfuerzo de flexión máximo permitido
P_c	Carga de pandeo
P	Carga a la que está sometida la viga
δ	Deflexión final de la viga
δ_{max}	Deflexión máxima permitida al final de la viga
h	Altura de la soldadura
ℓ	Longitud soldada de la viga

Símbolo	Término
<i>b</i>	Espesor de la viga
<i>t*</i>	La altura de la viga
<i>R</i>	Distancia desde el centroide del grupo de soldaduras hasta el punto en la soldadura de interés.
<i>M</i>	Momento
<i>J</i>	Momento polar de Inercia
<i>L</i>	Longitud de la viga en voladizo
<i>E</i>	Módulo de elasticidad del material
<i>G</i>	Módulo de rigidez del material
<i>psi</i>	Libras de fuerza por pulgada cuadrada
<i>lb</i>	Libras fuerza
<i>K'</i>	Constante para penalización estática
<i>m'</i>	Número de restricciones en el problema de optimización
<i>s'</i>	Número de restricciones del problema de optimización satisfechas

Introducción

Actualmente la sociedad está inmersa en el paradigma de la globalización, este ha creado un entorno de competencia en la cadena productiva; por lo tanto para ser efectivo en este ambiente, se requiere una alta calidad en los productos y un uso eficiente de los recursos.

Esta labor de producción eficiente recae en la ingeniería, sobre todo en el área del diseño. Por lo tanto, para lograr una mejora en este campo y adaptarse a las exigencias impuestas por la globalización, la ingeniería debe utilizar las herramientas a su disposición; una de estas es la optimización, la cual permite lograr el mejor resultado posible bajo ciertas circunstancias [1]. Sin embargo, esta tiende a ser bastante desafiante por la complejidad de los problemas de interés y también debido a las rigurosas normas de diseño en la práctica ingenieril [2].

Debido a tal desafío, la optimización ha estado sometida a un constante desarrollo, impulsado en gran medida, desde que el avance en la computación ha permitido la implementación y evaluación de algoritmos para realizar computaciones matemáticas de gran magnitud; con ello posibilitando resolver muchos problemas, que hasta entonces no tenían solución conocida [3]. De estos algoritmos computacionales se pueden destacar dos vertientes: los métodos de programación matemática y las técnicas de procesos estocásticos [1], siendo estas últimas donde se enmarcan las metaheurísticas, tema central de este documento.

De los métodos matemáticos basados en gradiente se puede resaltar que necesitan superficies suaves y funciones diferenciables [4]; condiciones que no se cumplen en todos los problemas, como ocurre en los no diferenciables, donde estos métodos aún no entregan resultados de calidad [5], [6], [7]. En cambio las técnicas estocásticas por lo general obtienen valores cercanos al óptimo en esos y otro tipo de problemas, sin necesidad de consideraciones especiales; razón por la cual la industria se hizo consiente del beneficio de las metaheurísticas y la importancia de explorar esta área [3].

Las técnicas de optimización metaheurística, funcionan por medio de la integración de aleatoriedad a procesos simples; orientando su búsqueda según las soluciones encontradas al evaluar la función objetivo [8]. Estas utilizan dos procesos principales al resolver problemas de optimización; la exploración y la explotación [9], [10]. La exploración se encarga de buscar globalmente soluciones prometedoras en el espacio de posibles respuestas; mientras que la explotación se ocupa de analizar localmente y en detalle esas soluciones prometedoras [9], [11].

Al indagar sobre las técnicas metaheurísticas, se ha encontrado que muchas de ellas son inspiradas en la naturaleza [2], [12], [13]; entendiendo por naturaleza cualquier parte del universo físico que no es producto del diseño humano [14]. Esta inspiración busca aprovechar características propias de los sistemas naturales, tales como: mecanismos de resolución de problemas, el aprendizaje, la memoria, la adaptación, la eficiencia, etc. Los cuales son atributos deseables en los algoritmos metaheurísticos [14].

Siendo entonces la naturaleza una fuente de inspiración fundamental para la creación de estas técnicas de optimización; en este trabajo se presenta una nueva técnica metaheurística, denominada algoritmo de optimización del pulpo artificial (AOOA por sus siglas en inglés), la cual emula el movimiento y algunas características anatómicas de los pulpos, posee nuevos procesos heurísticos y utiliza pocos parámetros de control. Esta es comparada, en términos de desempeño, con otras técnicas de optimización metaheurística y finalmente se emplea para la solución de un problema de diseño mecánico.

A continuación con el fin de complementar la introducción del lector a este trabajo, se lleva a cabo la presentación de los objetivos de la tesis y la organización del documento.

Objetivos de la Tesis

Objetivo General

- Desarrollar una técnica metaheurística de optimización inspirada en las características y movimiento de los pulpos.

Objetivos Específicos

Para garantizar un correcto desarrollo de este trabajo y lograr cumplir con el objetivo general, son necesarios los siguientes objetivos específicos:

- Diseñar e implementar un algoritmo metaheurístico de optimización inspirado en las características y movimiento de los pulpos.
- Validar el desempeño de la técnica metaheurística, mediante la evaluación de la solución de problemas de prueba.
- Analizar los resultados obtenidos y comparar el desempeño del algoritmo creado con el de otras técnicas metaheurísticas de optimización.
- Aplicar el algoritmo a un problema de diseño mecánico.

Organización de la Tesis

Este documento se compone de cinco capítulos y dos anexos. A continuación se da una descripción de cada una de estas partes.

El capítulo uno brinda una visión detallada de la optimización mono-objetivo, la cual incluye una clasificación de las técnicas disponibles en esta área, continúa con una definición de heurística y metaheurística, posteriormente se realiza un estado del arte de los algoritmos más representativos en el campo de la optimización metaheurística, e inmediatamente se describen brevemente los algoritmos desarrollados en la última década. Todo esto para dar paso a una descripción de las propiedades principales de las metaheurísticas.

El capítulo dos contiene una descripción de las características físicas de los pulpos; al ser este animal esencial para el desarrollo del trabajo, también se explican sus movimientos de desplazamiento y otras cualidades que son emuladas en el algoritmo; a la vez se describen las analogías empleadas para la creación de los operadores utilizados en la metaheurística y por último se realiza la descripción general del algoritmo.

El capítulo tres inicia listando los problemas de prueba utilizados en la evaluación de desempeño del algoritmo, a continuación se exponen los resultados de esta evaluación; posteriormente se lleva a cabo una comparación de rendimiento del algoritmo diseñado, frente a otras técnicas de optimización metaheurística. El capítulo finaliza con la selección adecuada de parámetros para AOOA con base en un análisis de sensibilidad de respuesta en función del valor de cada parámetro.

El capítulo cuatro incluye la solución, por medio de la metaheurística creada, del problema de optimización del diseño de una viga soldada. Además de la solución y análisis de resultados de este problema de diseño mecánico, se incluye una comparación con resultados obtenidos por medio de otras técnicas de optimización.

El capítulo cinco contiene las conclusiones resultado de la investigación y los trabajos futuros que se proponen con el desarrollo de esta tesis.

El anexo A presenta el algoritmo implementado en el Software MATLAB y El anexo B muestra los resultados de 30 ejecuciones del algoritmo para el problema de diseño de la viga soldada.

1. Marco Teórico

Este capítulo se encarga de abordar de manera clara y concisa temas básicos de optimización. En primer lugar se da una definición de ésta, se prosigue definiendo los términos utilizados en los problemas de optimización, para seguidamente exponer una clasificación de las técnicas existentes en optimización, consecutivamente se expone la definición de metaheurística. Posteriormente se realiza una presentación de los algoritmos más representativos en el campo de la optimización metaheurística, luego se realiza una breve descripción de las técnicas creadas en la última década; para finalmente plantear las características principales de éstas.

1.1 Conceptos generales de optimización

La optimización en su estado más simple, está relacionada con la mejora de cualquier actividad [15]. Una tarea cotidiana como lo es desplazarse de un lugar a otro, está sujeta a ser optimizada; por ejemplo, en función del tiempo. Ya sea seleccionando un camino más corto que el actual y por lo tanto más rápido, o incrementando la velocidad para disminuir la duración del recorrido. Se puede observar que la mejora en estos casos se realiza generalmente modificando los factores involucrados, como lo son las variables de diseño, en función de lo que desea optimizar; representado por la función objetivo. A continuación se procede a exponer la optimización en términos matemáticos, donde se definen conceptos fundamentales de ésta.

Definición 1 (*Problema de optimización*): un problema de optimización mono-objetivo está definido como la minimización (o maximización) de $f(\hat{x})$ [16], sujeto a las siguientes restricciones:

$$g_i(\hat{x}) \leq 0, i = \{1, \dots, m\} \quad (1.1)$$

Y a su vez sujeto a:

$$h_j(\hat{x}) = 0, j = \{1, \dots, p\} \hat{x} \in \Omega \quad (1.2)$$

La solución minimiza (o maximiza) la función escalar $f(\hat{x})$ donde \hat{x} es un vector n -dimensional de variables de decisión $\hat{x} = (x_1, x_2, \dots, x_n)$ de algún universo Ω .

A partir de la definición anterior se pueden precisar en forma detallada los siguientes conceptos:

Definición 2 (*Función Objetivo*): establece los criterios para evaluar si un diseño es mejor que otro; por lo general, es una función escalar que depende de las variables de diseño. Una solución óptima es aquella donde se minimiza o maximiza la función objetivo, de acuerdo con el caso deseado [17].

Asumir si maximizar o minimizar un problema, no restringe la generalidad de los resultados; puesto que se puede establecer una igualdad entre maximización y minimización de la siguiente forma [3], [18], [19]:

$$\max\{f(\hat{x})|\hat{x} \in \Omega\} \equiv \min\{-f(\hat{x})|\hat{x} \in \Omega\} \quad (1.3)$$

Definición 3 (*variables de diseño o decisión*): son las cantidades variables del problema de optimización que conforman el vector \hat{x} [1]; sus valores deben ser determinados para resolver el problema de optimización. Pueden ser continuas o discretas y sus valores varían durante el proceso de optimización del algoritmo [17].

Definición 4 (*restricciones*): son las condiciones aplicadas a las variables, se utilizan para especificar un conjunto factible de puntos \hat{x} en el espacio \mathbb{R}^n [4]. Nótese que las ecuaciones (1.1) y (1.2) muestran las restricciones que se deben cumplir.

Definición 5 (*Parámetros del problema*): son magnitudes que permanecen fijas durante el proceso de optimización. Por lo general responden a las condiciones impuestas en el problema desde el exterior [17].

Definición 6 (*Espacio de búsqueda*): es el espacio Ω que contiene todos los valores posibles para las variables de decisión x .

1.2 Clasificación de técnicas en optimización

Una clasificación simple para las técnicas, es la de dividir las en programación matemática y metaheurísticas. Algunas de las técnicas de programación matemática, como las fundamentadas en gradiente, garantizan hallar la solución óptima; el problema de estas es que necesitan ciertas condiciones como: superficies suaves y funciones diferenciables [4], [20]. De las metaheurísticas se puede mencionar que utilizan pocos recursos y sin necesidad de condiciones especiales en la función objetivo, encuentran generalmente soluciones cercanas a la óptima; razón por la que están recibiendo atención en la comunidad internacional [5], [7], [13], [21].

En la Figura 1.1, modificada de Luna [21], se observa una clasificación de las técnicas de optimización; allí se muestra que las técnicas metaheurísticas se dividen en dos clases: heurísticas ad hoc y metaheurísticas. Dentro de las heurísticas ad hoc, están los procedimientos constructivos y los de búsqueda local. Los primeros construyen la solución definiendo diferentes partes de ella a partir de pasos sucesivos [22], estos suelen ser veloces en hallar una solución; sin embargo las respuestas halladas, son por lo general de baja calidad [21]. Los segundos parten de una solución conocida y recorren el espacio de búsqueda esperando encontrar un óptimo local, prosiguen explorando las posibles soluciones en los vecindarios, para ir mejorando la solución obtenida; en estos se hace inabordable evaluar todas las soluciones existentes en el espacio de búsqueda [21]. Las metaheurísticas, se pueden dividir en dos tipos, de trayectoria y con base en población.

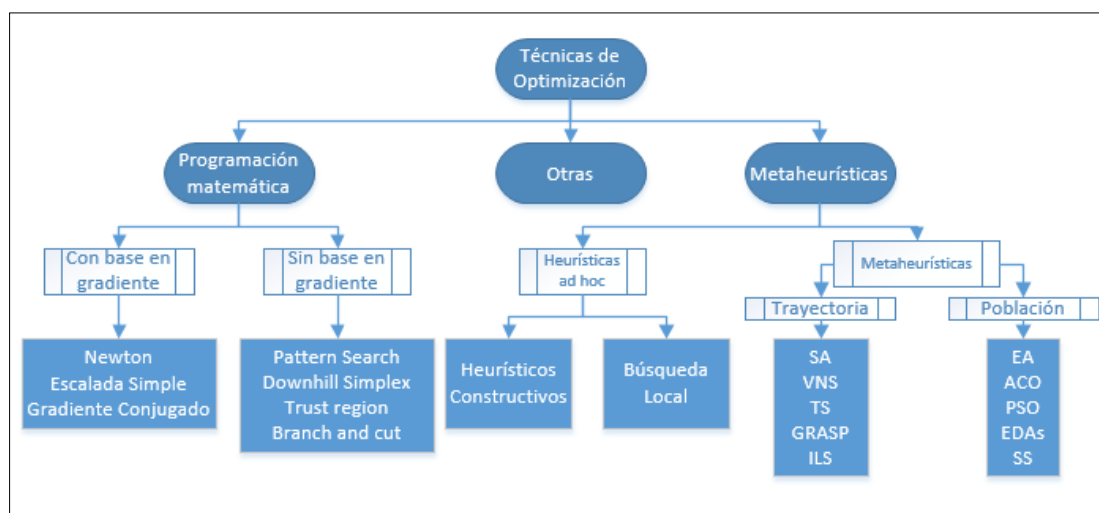


Figura 1.1 Clasificación de las técnicas de optimización, modificada de [21].

1.3 Heurística y metaheurística

Para lograr un entendimiento global de lo que el término metaheurística abarca, es necesario, en principio, conocer los términos griegos que la forman. En primer lugar está el prefijo "meta" el cual significa de "nivel superior" o "más allá"; en segundo lugar se tiene "heurística" que expresa "descubrir" o "guiar una investigación" [23]. Con base en lo anterior se puede definir que una metaheurística es: un proceso superior de investigación. Con el fin de profundizar en esta idea, se brinda a continuación una definición formal de la palabra heurística.

Definición 7 (*heurística*): "Procedimiento simple, a menudo fundamentado en el sentido común, que se supone ofrecerá una buena solución a problemas difíciles, de un modo fácil y rápido"[23, p.2].

De lo anterior se puede concluir que una heurística, es un procedimiento de búsqueda que se fundamenta en la aplicación de ciertas reglas sencillas y que no garantiza la obtención del óptimo global. La metaheurística al ser un proceso de alto nivel, utiliza a las heurísticas para realizar una exploración efectiva del espacio de búsqueda y obtener mejores resultados. Pueden definirse como un conjunto de estrategias inteligentes para mejorar la eficiencia de los procedimientos heurísticos [12]. A partir de estos conceptos previos, se puede dar a continuación una definición formal de metaheurística.

Definición 8 (*metaheurística*): "Una metaheurística es un marco de trabajo algorítmico de un problema independiente de alto nivel, que proporciona un conjunto de directrices o estrategias para desarrollar algoritmos de optimización heurística"[25, p.4].

1.4 Algoritmos representativos de optimización metaheurística

A continuación se procede a realizar una descripción general de algunos algoritmos metaheurísticos de optimización; estos se encuentran listados cronológicamente con el fin de mostrar los inicios de este campo de investigación y su respectiva evolución.

1.4.1 Algoritmo genético (Genetic Algorithm-GA)

En el año 1975, Holland [25] introduce el algoritmo genético, la idea general de este algoritmo es imitar un proceso natural, en este caso la evolución biológica. Esto lo realiza a

partir de ciertas operaciones que simulan los procesos que allí ocurren. Se empieza con una población aleatoria de individuos, los cuales representan las variables iniciales, estos son evaluados en la función objetivo y a partir de los resultados obtenidos se valora la aptitud de cada individuo, esto se realiza al establecer un factor de adecuación conocido como “fitness”. Este factor determina qué individuos arrojaron mejores resultados; en el caso de la minimización, serán los individuos con valores más bajos conseguidos en la función objetivo. La anterior operación posee el nombre de evaluación.

La siguiente fase, denominada selección, es realizada con base en la adecuación a la función objetivo; ya que determina qué tan probable será la elección de un individuo, para que actúe como padre en la próxima generación. El anterior proceso de selección puede ser realizado por medio de ciertas estrategias, como lo son: selección por ruleta, selección por rango, selección por torneo, selección elitista y selección por clasificación [5], [18].

El siguiente operador denominado cruce, el cual simula la reproducción biológica, es utilizado para transformar y transmitir la información contenida en los padres a cada individuo; para esto el valor de cada individuo se transforma a base binaria, donde cada 1 o 0 representan un gen del cromosoma o valor del individuo. Posteriormente se toman dos individuos seleccionados como padres y se entremezclan partes de su información por medio de la combinación de genes, según algún criterio determinado. Luego de la mezcla se obtienen dos nuevos individuos que conservan características compartidas de ambos padres [26].

Durante el cruce, el resultado de la reproducción puede ser afectado por un operador conocido como mutación. Durante su ejecución este puede modificar aleatoriamente uno o varios genes de los nuevos cromosomas; alterando de esta forma los individuos obtenidos.

Al haber utilizado los cuatro operadores anteriores sobre la población inicial, se obtienen los nuevos individuos; los cuales son sometidos al mismo proceso, hasta que se cumple algún criterio de parada definido por el usuario [27].

El algoritmo genético es una técnica fácil de entender y prácticamente no exige un conocimiento matemático avanzado. Permite resolver diferentes tipos de problemas de optimización y es fácil de combinar con otros algoritmos. Sus debilidades radican en que utiliza operadores complejos para la selección y cruce, además tiene una tasa de convergencia prematura [12].

1.4.2 Algoritmo de la colonia de hormigas (Ant Colony Optimization-ACO)

Dorigo en el año 1992 propone, en su tesis de doctorado, el algoritmo de la colonia de hormigas; posteriormente lo publica en 1999 [28]. Este algoritmo se fundamenta en la conducta de las hormigas cuando viajan al exterior del nido en busca de comida. El objetivo de la técnica metaheurística es buscar rutas óptimas para realizar este viaje [29].

En la naturaleza cuando las hormigas salen en busca de alimento, en principio deambulan aleatoriamente hasta que lo encuentran; cuando eso sucede emprenden su viaje de vuelta al nido. Durante este recorrido de vuelta, inician un proceso bioquímico de dejar feromonas a su paso; que al ser atrayentes para todas las hormigas, permiten que otras probablemente sigan el mismo trayecto y al regresar con comida, intensifiquen a su vez la cantidad de feromonas que se encuentran en este camino [28], [30].

No obstante, las feromonas se evaporan con el transcurrir del tiempo, lo que hace menos llamativo tal recorrido, permitiendo que únicamente los caminos cortos al objetivo donde transitan constantemente estos animales, sean las vías predilectas; logrando al final transitar únicamente por el trayecto más breve posible. El camino más corto representa en el algoritmo la convergencia al óptimo global [31].

En la implementación se simula ese comportamiento, introduciendo hormigas virtuales en un grafo que constituya el problema en cuestión. Esto se hace al utilizar K hormigas que iterativamente pasan de un estado “ y ” a otro “ z ”; las hormigas en cada iteración calculan un conjunto de posibles movimientos y se desplazan a uno de los estados por medio de una probabilidad. Para una hormiga K la probabilidad de moverse de un estado “ y ” a uno “ z ”, se determina por P_{yz}^k ecuación (1.4), en la cual η_{yz} mide que tan atractivo es el movimiento a dicha posición y τ_{yz} la conveniencia de dicho movimiento en función de que tan favorable ha sido ese movimiento en ocasiones pasadas; α y β son parámetros para controlar la influencia de esos dos factores [28], [31].

$$P_{yz}^k = \frac{\tau_{yz}^{\alpha} * \eta_{yz}^{\beta}}{\sum \tau_{yz}^{\alpha} * \eta_{yz}^{\beta}} \quad (1.4)$$

Posterior a este proceso, después de que cada hormiga ha realizado su cambio de estado; es decir a encontrado una solución, se actualizan los valores de τ_{yz} .

Las características principales de este algoritmo son: su difícil codificación y su alta eficacia en problemas de tipo combinatorio [20].

1.4.3 Optimización por enjambre de Partículas (Particle Swarm Optimization-PSO)

En el año 1995 Kennedy & Eberhart [32] formulan la optimización por enjambre de partículas. Esta técnica emula un comportamiento social característico de organismos en movimiento, como una bandada de aves, un banco de peces, enjambres de abejas, etc. Aquel comportamiento social es la inteligencia de manada; la cual se evidencia cuando los individuos se comunican entre sí con alguna finalidad. En este caso el propósito es informarse acerca de posiciones benéficas en el espacio.

Al simular este comportamiento, se establecen varios individuos con posiciones aleatorias, los cuales se representan en el algoritmo por medio de partículas en el espacio de búsqueda; con el transcurrir de cada iteración, las partículas se trasladan de su posición actual a otra. El cambio de posición está influenciado por la velocidad determinada por la ecuación (1.5), en la que influye la experiencia propia de la partícula (mejor posición de ella a través del tiempo) y la experiencia colectiva (posición de la mejor solución candidata a través del tiempo). La posición de la partícula (\hat{y}) se determina por medio de la ecuación (1.6) [7].

$$\hat{v}^t = w * \hat{v}^{t-1} + c_1 \cdot r_1 \cdot (\hat{b} - \hat{y}^{t-1}) + c_2 \cdot r_2 \cdot (\hat{g} - \hat{y}^{t-1}) \quad (1.5)$$

$$\hat{y}^t = \hat{y}^{t-1} + \hat{v}^t \quad (1.6)$$

Donde w es la inercia, r_1 y r_2 son términos aleatorios, c_1 y c_2 son valores de aceleración, \hat{v} es la velocidad de la partícula, \hat{b} es la mejor posición propia y \hat{g} es la mejor posición global.

Las ventajas que identifican a este algoritmo son: una estructura simple que permite una fácil implementación y una alta rapidez para obtener resultados; sin embargo, tiende a quedarse atrapado en óptimos locales, al enfrentarse a problemas multimodales [31].

1.4.4 Evolución diferencial (Differential Evolution-DE)

Storn y Price en el año 1997 presentan la evolución diferencial [33]; una técnica estocástica directa de búsqueda global, la cual está relacionada con los algoritmos de tipo evolutivo.

Para su funcionamiento esta técnica utiliza NP vectores como población. Durante la ejecución del algoritmo, tales vectores son sometidos durante cada generación a tres procesos: mutación, cruce y selección, los cuales son descritos a continuación.

La mutación en este algoritmo, crea un vector perturbado $\hat{\mathbf{m}}_{j,i}$ con $i = 1, 2, \dots, NP$ y $j = 1, 2, \dots, D$; para ello suma a un vector $\hat{\mathbf{x}}_{j,b_1}$ la diferencia ponderada de los vectores $\hat{\mathbf{x}}_{j,b_2}$ y $\hat{\mathbf{x}}_{j,b_3}$; como se observa en la ecuación (1.7). Donde los índices $b_1, b_2, b_3 \in \{1, 2, \dots, NP\}$, son diferentes entre sí y elegidos aleatoriamente. El factor F es un valor real y constante, perteneciente al intervalo $[0, 2]$, que controla la amplificación de la diferencia entre los vectores $\hat{\mathbf{x}}_{j,b_2}$ y $\hat{\mathbf{x}}_{j,b_3}$.

$$\hat{\mathbf{m}}_{j,i} = \hat{\mathbf{x}}_{j,b_1} + F \cdot (\hat{\mathbf{x}}_{j,b_2} - \hat{\mathbf{x}}_{j,b_3}) \quad (1.7)$$

El algoritmo continúa con el proceso de cruce, el cual mezcla el vector obtenido en la mutación $\hat{\mathbf{m}}_{j,i}$, con un vector seleccionado de la población, denominado vector objetivo $\hat{\mathbf{t}}_{j,i}$; cada miembro de la población es seleccionado como vector objetivo, únicamente una vez por generación. Esta combinación de vectores se observa en la ecuación (1.8); de donde resulta el vector de prueba $\hat{\mathbf{e}}_{j,i}$.

$$\hat{\mathbf{e}}_{j,i} = \begin{cases} \hat{\mathbf{m}}_{j,i} & \text{sí } d \leq CR \text{ or } j = b_4 \\ \hat{\mathbf{t}}_{j,i} & \text{sí } d > CR \text{ and } j \neq b_4 \end{cases} \quad (1.8)$$

Donde d es un número aleatorio perteneciente al intervalo $[0, 1]$, CR es la constante de cruce, la cual se escoge en el intervalo $[0, 1]$ y b_4 es un índice aleatorio que pertenece al conjunto $\{1, 2, \dots, D\}$. Por último se evalúan en la función objetivo a los vectores $\hat{\mathbf{e}}_{j,i}$ y $\hat{\mathbf{t}}_{j,i}$ de los cuales se selecciona el que mejor desempeño haya obtenido; con el fin de utilizarlo como integrante de la población en la próxima generación.

Dentro de sus características principales se encuentra su sencillez para codificar, su propiedad para converger, la cual hace que la población aumente su avance en funciones planas o de valles superficiales y que disminuya su paso en zonas cercanas al mínimo; es decir su habilidad para explorar rápidamente y explotar zonas de interés [33].

1.4.5 Algoritmo de optimización del forrajeo de bacterias (Bacterial Foraging Optimization Algorithm-BFOA)

En el año 2002 Passino [34] propone el algoritmo BFOA, que emula el comportamiento y algunas características de las bacterias cuando buscan alimento. Específicamente la fuente de inspiración es la bacteria *E. Coli* y la táctica que ella emplea con el fin de obtener nutrientes del ambiente para sobrevivir y reproducirse.

Para lograr imitar esa habilidad se utilizan en el algoritmo analogías de la posición y el movimiento de la bacteria, eventos de eliminación y dispersión de este organismo, la reproducción del mismo y fenómenos de atracción y dispersión de congéneres en el área circundante; entre otros.

Al emular la estrategia de la bacteria, se realiza una analogía entre el espacio de búsqueda y la zona donde se distribuyen los nutrientes. Esto se realiza por medio de la definición de una función $J^*(\theta)$ con $\theta \in \mathbb{R}^n$. Siendo θ la bacteria y $J^*(\theta)$ la superficie de nutrientes del ambiente. La función $J^*(\theta)$ puede tomar valores como por ejemplo:

$J^*(\theta) < 0$ Implicando que la bacteria se encuentra en un ambiente rico en nutrientes.

$J^*(\theta) = 0$ Implicando que la bacteria se encuentra en un ambiente neutral.

$J^*(\theta) > 0$ Implicando que la bacteria se encuentra en un ambiente toxico o perjudicial.

La bacteria se desplaza buscando zonas de mayor concentración de nutrientes, es decir en la dirección decreciente de $J^*(\theta)$, buscando el mínimo costo de la función por medio de una caminata aleatoria con cierto sesgo [34]. Esta marcha se compone de nado constante para avanzar y giro para cambiar la orientación. Biológicamente esto lo realiza gracias a los flagelos de su cuerpo, los cuales usa para alternar su movimiento y orientarse como respuesta a un estímulo químico denominado quimiotaxis [35].

Con base en los movimientos de nado y giro, se define el paso quimiotáxico, el cual emula el avance que realiza la bacteria secuencialmente para trasladarse; ya sea un giro seguido por un giro o un giro seguido por nado lineal. Al implementar la metáfora, son necesarios además del paso quimiotáxico, el índice de reproducción k^* , eventos de dispersión y eliminación l^* , el índice del paso quimiotáxico j^* , P' la posición de cada bacteria definida por la ecuación (1.9), la población T^* , el costo de la localización de la bacteria $J^*(\theta)$ y la vida de la bacteria medida en cierta cantidad N_c de pasos quimiotáxicos.

$$P'(j^*, k^*, l^*) = \{\theta^i(j^*, k^*, l^*) | i = 1, 2, \dots, T^*\} \quad (1.9)$$

Para dimensionar estos movimientos de nado y giro, se define un tamaño de paso $C(i)$ mayor que cero y una dirección aleatoria $\varphi(j)$; este paso se utiliza para simular el avance de la bacteria hacia una posición diferente, ver ecuación (1.10). En caso de que la nueva posición ostente un costo menor al evaluarse en la función objetivo, se continua nadando en esta dirección hasta cumplir un predefinido número de pasos N_s ; momento en el cual se define nuevamente una dirección aleatoria hacia la cual avanzar.

$$\theta^i(j^* + 1, k^*, l^*) = \theta^i(j^*, k^*, l^*) + C(i)\varphi(j) \quad (1.10)$$

Es implementado adicionalmente en el algoritmo, un comportamiento de tipo social que hace a las bacterias utilizar atrayentes o repelentes entre sí; propiciando la comunicación entre ellas y el desplazamiento como grupo.

La reproducción ocurre cada cierta cantidad N_c de pasos quimiotáxicos, el proceso se realiza duplicando a la mitad de bacterias con menor costo acumulado $J^*(\theta)$, las copias de los individuos iniciales se ubican en la misma posición que estos; aumentado así en el algoritmo las posiciones benéficas con las que trabajar. El resto de los organismos son eliminados para mantener la población constante, a medida que avanzan las iteraciones. Para finalizar la implementación, se utilizan los eventos de eliminación y dispersión, los cuales pueden ocurrirle a cualquier bacteria, según una probabilidad P_{ed} .

Este algoritmo utiliza numerosos parámetros y operaciones, haciendo que sea un algoritmo de excesiva extensión y de configuración compleja para algunos usuarios; siendo por lo tanto una técnica demorada en su ejecución y codificación [36].

1.4.6 Algoritmo de colonia artificial de abejas (Artificial Bee Colony-ABC)

Karaboga plantea en el año 2005 [37] y luego detalla en el 2007 [38] en compañía de Basturk, la técnica metaheurística ABC; la cual emula una colonia de abejas. Este algoritmo considera la búsqueda de comida de las abejas artificiales, como un símil a la búsqueda de posiciones benéficas en el espacio de búsqueda considerado. Para ello dentro de la colonia artificial, existen tres tipos de abejas: las empleadas BN , las observadoras y las

exploradoras; cada una de ellas realiza una labor diferente y todas trabajando en conjunto, son la esencia de este algoritmo.

Esta metaheurística en su primera fase, utiliza a las abejas empleadas; enviándolas a posiciones aleatorias dentro del espacio de búsqueda. Sus posiciones son entonces evaluadas en la función objetivo y según su adecuación a esta, se determina cuales encontraron mejores fuentes de comida. Seguidamente esta información es comunicada a las abejas observadoras, con el fin de influenciar la decisión sobre el lugar al cual irán estas a realizar su búsqueda de alimento. Esta decisión la realizan según una probabilidad asociada al “fitness” de las abejas trabajadoras, según se indica en la ecuación (1.11); donde el número de fuentes de comida SN es igual al número de abejas empleadas y fit_i es el valor de adecuación en la función objetivo de la abeja considerada.

$$P_i^* = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad (1.11)$$

Posterior a esa comunicación, las abejas empleadas salen a buscar una nueva posición; esta se encuentra asociada a la información de la ubicación previa que poseen en su memoria y la obtienen según la ecuación (1.12). Donde \hat{x} es su posición previa, i y $b_1 \in \{1, 2, \dots, BN\}$, b_1 es aleatorio y diferente de i , φ es un valor real y constante perteneciente al intervalo $[-1, 1]$ y j es determinado por la dimensión del problema.

$$\hat{c}_{i,j} = \hat{x}_{i,j} + \varphi_{i,j}(\hat{x}_{i,j} - \hat{x}_{b_1,j}) \quad (1.12)$$

Las abejas observadoras utilizan la ecuación (1.12), para explotar las fuentes de comida que fueron elegidas gracias a la probabilidad dada por la ecuación (1.11), junto a un proceso de rueda de ruleta, selección por torneo o algún otro sistema de elección. Tanto en el caso de las abejas empleadas y las abejas observadoras, si la posición no puede ser mejorada por medio de la ecuación (1.12), se conserva la posición previa; es decir el algoritmo utiliza selección ávida.

Por último, el algoritmo establece que si la posición de una abeja no mejora después de cierto número límite de iteraciones, se abandone la fuente de alimento y sea remplazada por la posición aleatoria de una abeja exploradora.

Este algoritmo se caracteriza por su fácil implementación y su robustez; puede ser utilizado en gran variedad de problemas de optimización sin mayores modificaciones y posee pocos

parámetros de control, en comparación a otras técnicas [38], [39]. La desventaja primordial de esta metaheurística, es la inclusión de un factor de fitness, que genera lentitud cuando se utiliza en procesamiento serial; además de necesitar una alta cantidad de evaluaciones de la función objetivo [39], [40].

1.4.7 Algoritmo de luciérnagas (Firefly Algorithm-FA)

En el 2008 Yang [41] expone el algoritmo de luciérnagas. Esta técnica metaheurística se inspira en las señales ópticas de luz intermitente que se envían entre si las luciérnagas en las noches de verano.

Las bases de este algoritmo son sencillas, en primer lugar, las luciérnagas poseen un solo sexo; por lo tanto cualquiera de ellas puede ser atraída hacia la luz emitida por otra de las presentes. En segundo lugar, la intensidad del brillo determina qué luciérnaga es atraída a otra; ya que las luciérnagas con menos brillo, se ven atraídas hacia las que poseen uno mayor. En tercer lugar está establecido que la seducción entre insectos, se relaciona de manera inversa a su separación. Como última regla se tiene que si no existe una luciérnaga más brillante, todas se mueven aleatoriamente [41].

Para emular tal comportamiento, se utiliza la ecuación (1.13), la cual muestra cómo se actualiza la posición para cualquier luciérnaga. Donde β^* es el factor de atracción entre luciérnagas y q es la distancia entre ellas, α es un parámetro que controla el tamaño de paso y ϵ es un vector dibujado desde una distribución de Gauss.

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \beta^* \cdot e^{-\epsilon \cdot q_{ij}^2} * (\mathbf{X}_j^t - \mathbf{X}_i^t) + \alpha_t \epsilon_t \quad (1.13)$$

Este algoritmo posee una versión discreta, la que puede resolver eficientemente problemas de secuenciación de tareas en sistemas de producción, catalogados como NP-hard [42].

1.5 Algoritmos establecidos en la última década

Este campo de investigación ha mostrado un crecimiento exponencial en los últimos años; evidente por la abundante cantidad de nuevos algoritmos metaheurísticos publicados. Realizar un resumen detallado de cada uno de ellos no es el objetivo de este documento; por ello únicamente se realiza una breve descripción de algunas técnicas metaheurísticas de los últimos años.

1.5.1 Algoritmo de optimización reacción química artificial (Artificial Chemical Reaction Optimization Algorithm-ACROA)

En el año 2011, Alatas presenta el ACROA [13], inspirado en reacciones químicas las cuales simula de manera artificial. Para la implementación se considera un recipiente de volumen fijo, el cual posee en su interior cierta cantidad inicial N^* de reactivos; estos interactúan constantemente entre sí, según unas normas predeterminadas que emulan las reacciones químicas. A partir de estas reacciones, se obtienen nuevos reactivos (soluciones candidatas), los cuales se evalúan en la función objetivo, con el fin de determinar los mejores reactivos a incluir en el conjunto de reactantes, para interactuar en las siguientes iteraciones. Este proceso continúa hasta cumplir un criterio de parada, momento en el cual no pueden suceder más reacciones químicas, el cual es un símil de una solución inerte.

El autor establece que el algoritmo posee la habilidad, para realizar a la vez búsqueda global y local; por lo tanto no necesita un sistema independiente de búsqueda local para refinar sus respuestas. Destaca además entre sus fortalezas, que ACROA no utiliza una función extra de relación como lo es el “*fitness*”, para determinar la calidad de los reactivos. Expresa también que el algoritmo no requiere demasiados parámetros a especificar [13].

1.5.2 Algoritmo de optimización con base en la enseñanza y el aprendizaje (Teaching-Learning-Based Optimization-TLBO)

En el año 2011 Rao y otros, formulan la técnica metaheurística TLBO [43], la cual es una técnica para optimizar problemas de diseño mecánico con restricciones; este algoritmo se encuentra inspirado en la influencia que tiene un maestro en un salón lleno de aprendices. Esta metaheurística es de tipo poblacional; en la cual las soluciones candidatas son emuladas como un grupo de alumnos en un salón de clase. TLBO posee dos fases, la primera es la etapa del maestro, en la cual se toma la mejor solución candidata para influenciar el cambio en la población de aprendices; en la segunda parte, los estudiantes aprenden por interacción entre ellos, siguiendo siempre al mejor estudiante del aula.

Entre sus características se destaca el uso de parámetros aleatorios internos. Según sus autores, esta técnica se desempeña mejor y con menor esfuerzo computacional en problemas de alta dimensión que sus contendores, entre ellos ABC [38], PESO [44] y CDE [45].

1.5.3 Algoritmo de pastoreo de Kril (Krill Herd-KH)

En el año 2012, Gandomi y Alavi formulan el algoritmo KH [46], el cual es un algoritmo metaheurístico que emula el comportamiento de pastoreo de los individuos de kril; tanto en su habilidad para encontrar comida, como en su capacidad de ser influenciados por la manada. Este algoritmo que idealiza el comportamiento del kril, utiliza la distancia del mismo con respecto a la comida como un símil a la función objetivo. Para ello formula la posición del individuo de kril según tres factores: primero el movimiento inducido por la presencia de otros individuos de kril, segundo la actividad de forrajeo del individuo y por último considera un factor de difusión aleatoria.

Dentro de sus propiedades se destacan el uso de dos técnicas globales y dos locales; además emplea los operadores genéticos mutación y cruce, con el fin de obtener mayor precisión en los resultados. Los autores destacan que cada individuo de kril, aporta al proceso según su “fitness”; también realzan los efectos de atracción o repulsión que ejercen en sus vecinos, los cuales producen un efecto de búsqueda local. Recalcan la habilidad del algoritmo para determinar la posición del alimento, la cual se estima en la ubicación del mejor valor global. El algoritmo fue comparado con técnicas metaheurísticas como GA, ES, ACO, PSO, entre otras; los resultados obtenidos por sus autores muestran que el algoritmo KH supera a estas conocidas técnicas.

1.5.4 Algoritmo del ciclo del agua (Water Cycle Algorithm-WCA)

En el año 2012 Eskandar y otros exponen el WCA [47]; este algoritmo está inspirado en el ciclo hidrológico y en como fluyen los arroyos y ríos colina abajo hacia el mar. Este algoritmo es aplicado a optimizar problemas de diseño en ingeniería con restricciones. La metaheurística es de tipo poblacional; para su ejecución se considera una precipitación inicial para la generación de sus individuos, los cuales se denominan: gotas de lluvia. Posteriormente estas son seleccionadas como el mar, ríos o arroyos; según corresponden el mejor, intermedios y peores valores de adecuación en la función objetivo. Luego los arroyos fluyen a los ríos y a su vez los ríos al mar, según una función de acercamiento; también se incluye un operador denominado evaporación, el cual realiza una función similar a la mutación en los algoritmos genéticos.

Según el estudio comparativo realizado por sus autores, el algoritmo generalmente ofrece mejores soluciones, en cuanto a precisión y menor esfuerzo computacional, que otros algoritmos de optimización considerados, como lo son ABC [38], GA [25] y TLBO [43], entre otros. Lo que lo hace un algoritmo competitivo; sin embargo la calidad de las soluciones y la eficiencia computacional están sujetas a la naturaleza y complejidad del problema [47].

1.5.5 Algoritmo de optimización búsqueda de pingüinos (Penguins Search Optimization Algorithm-PeSOA)

En el año 2013 Gheraibia y Moussaoui plantean el PeSOA [48], una técnica estocástica que permite resolver problemas de optimización de tipo continuo. Este algoritmo de tipo poblacional, emula la estrategia colaborativa de caza de los pingüinos. Para su implementación se considera que la población, está compuesta por grupos de cierto número variable de pingüinos. Estos grupos cazan en locaciones distintas y al salir del agua comparten la información de posiciones benéficas a su grupo; con el fin de que este se vea favorecido en la cacería. Posteriormente estos grupos comparten la información entre ellos, con el fin de encontrar el mejor grupo y realizar una nueva distribución de pingüinos en el área a explotar.

Los autores recalcan la comunicación intragrupal e intergrupala, cualidades que permite al algoritmo realizar una búsqueda eficiente. Entre sus fortalezas se destaca que el algoritmo puede detectar todos los mínimos locales y el mínimo global; siempre que la población sea lo suficientemente grande. Expresan además sus creadores que PeSOA es más robusto y eficiente que otros algoritmos metaheurísticos, ya que su estrategia de búsqueda no recae en cambiar la posición del mejor encontrado, sino en el trabajo por grupos de individuos.

1.5.6 Algoritmo de optimización del arrecife de coral (The Coral Reefs Optimization-CRO)

En el año 2014 Salcedo y otros desarrollan el CRO [49], el cual es un algoritmo que simula artificialmente un arrecife de coral. Este considera diferentes especies de coral (soluciones del problema); utiliza características como la reproducción, ya sea de tipo sexual o asexual, la depredación y las batallas entre congéneres.

La técnica emula a este invertebrado, creando un espacio reticular A de tamaño $M^* \times N$; cada una de las posiciones i, j de la matriz, representa un espacio que puede alojar a una especie de coral. Inicialmente se realiza una distribución aleatoria de especies de coral en algunos de los espacios existentes, dejando a la vez otros lugares vacíos. A continuación se evalúa la salud del coral; la cual es una función de adecuación asociada a la función objetivo. Posteriormente se realiza la reproducción, en el caso de ser sexual, se realiza como un cruce entre dos especies de coral, las cuales pueden ser escogidas aleatoriamente o con otro tipo de estrategias de selección; en el caso de ser reproducción asexual, se puede dar como una copia de la especie (solución actual) o como una mutación de la misma. Los descendientes se alojan en los espacios vacíos circundantes, permitiendo la expansión de la colonia; no obstante en caso de no haber espacios libres, se lucha por apoderarse de ellos. Labor que se realiza por medio de la comparación de la salud del coral; es decir su valor de adecuación a la función objetivo. Finalmente, algunas de las especies mueren según una probabilidad; este procedimiento se realiza hasta que se cumple algún criterio de parada.

Los autores mencionan que este algoritmo posee buenas propiedades de convergencia al óptimo global; además los prometedores resultados obtenidos en su investigación, al aplicar el algoritmo a un caso de despliegue de redes móviles en España y el diseño óptimo de una granja eólica en el norte de Europa, incentiva la aplicación de CRO a problemas reales de alta complejidad en optimización.

1.5.7 Algoritmo búsqueda de organismos simbióticos (Symbiotic Organisms Search-SOS)

En el año 2014, Cheng y Prayogo formulan el SOS [50], un algoritmo inspirado en las interacciones que se producen entre organismos simbióticos en la naturaleza. SOS emplea una población inicial que denomina organismos, los cuales son sometidos a tres interacciones; las cuales son similares al mutualismo, el comensalismo y el parasitismo. El mutualismo hace que dos individuos influyeran entre sí sus nuevas posiciones en el espacio de búsqueda; el comensalismo hace que uno de sus organismos, vea influenciada su ubicación en el espacio de búsqueda, gracias a otro organismo el cual no se ve afectado; por último, el parasitismo hace que un nuevo individuo creado aleatoriamente elimine a alguno de los organismos existentes; cabe resaltar que en estos procesos, los nuevos valores hallados son aceptados únicamente si son mejores que los anteriores.

Los creadores señalan que los tres operadores son simples operaciones matemáticas, las cuales son sencillas de utilizar; además en estos operadores no se incluye ningún parámetro de ajuste, con lo que se le da estabilidad al algoritmo. Generando por lo tanto un algoritmo robusto y sencillo de implementar.

1.5.8 Algoritmo de optimización del alga artificial (Artificial Algae Algorithm-AAA)

En el año 2015 Uymaz y otros proponen el AAA [51], esta técnica se inspira en los comportamientos de vida de las microalgas; específicamente en las especies fotosintéticas. Esta técnica define al óptimo global, como la posición donde la alga recibe la mejor cantidad de luz para su proceso fotosintético. Para lograr tal posición se inspira en procesos evolutivos, adaptativos y a su vez en el movimiento de este organismo; de allí establece tres operadores para la obtención de nuevas soluciones. El primero la reproducción, la cual es de tipo mitótico; por lo tanto si el alga virtual posee una adecuación alta en la función objetivo, se producen dos individuos idénticos. El proceso adaptativo hace que las algas con la peor adecuación a la función objetivo, se adecuen al ambiente, al emular a las algas en las mejores posiciones; es decir, se desplacen a una posición similar al individuo con mejor valor en la función objetivo. El tercer proceso de la metaheurística, es un movimiento de tipo helicoidal, el cual simula el comportamiento de las algas al acercarse a la superficie del agua, con el fin de recibir mayor cantidad de luz para su proceso fotosintético; este se emula en el algoritmo al cambiar la posición del alga en tres dimensiones aleatoriamente seleccionadas. Proceso realizado por medio de una interacción con otras algas seleccionadas por torneo y considerando la fuerza de arrastre del agua como un factor que influencia el resultado de las nuevas posiciones.

El algoritmo utiliza un proceso de selección codiciosa para aumentar la velocidad de convergencia; que junto a la diversidad obtenida gracias a la selección por torneo, consigue según sus autores un algoritmo estable y flexible. AAA posee tres parámetros de sintonización, los cuales son sensibles a producir que el algoritmo quede atrapado en mínimos locales o una convergencia prematura, si no se configuran correctamente. Al evaluar su desempeño genero resultados exitosos y balanceados en varias funciones de prueba con diferentes dimensiones. Según el “Wilcoxon signed-rank test” [52], AAA mostró estadísticamente ser más exitoso que ABC [38], DE [33] y ACO [53].

1.5.9 Algoritmo de optimización de la ballena (The Whale Optimization Algorithm-WOA)

En el año 2016 Mirjalili y Lewis establecen el WOA [11]. Este copia el comportamiento social de las ballenas jorobadas al momento de cazar; concretamente se inspira en la estrategia conocida como red de burbujas. El algoritmo emplea tres operadores: rodear la presa, mecanismo circundante de encogimiento y actualización de posición en espiral.

WOA asume que la posición de la presa acorralada por las yubartas, es la mejor solución candidata; por lo tanto es el objetivo en torno al cual las ballenas artificiales se acomodan. Para el uso de la estrategia red de burbujas se tienen los dos operadores antes mencionados, el encogimiento y el desplazamiento en espiral; el primero reacomoda las ballenas en posiciones que reducen la distancia al objetivo, el segundo hace que las ballenas se acerquen a la presa con un movimiento en espiral. Estos dos últimos procesos ocurren alternadamente por iteración, según una probabilidad de 50% para cada uno de ellos.

Este algoritmo fue sometido a un extensivo estudio comparativo, donde se puso a prueba con veintinueve funciones de prueba. En este se analizó la capacidad de exploración, explotación, el comportamiento de convergencia y la capacidad para escapar de mínimos locales del algoritmo. En tal comparación WOA fue hallado competitivo al cotejarlo con otras técnicas metaheurísticas [11].

1.5.10 Algoritmo de optimización del saltamontes (Grasshopper Optimisation Algorithm-GOA)

Saremi y otros en el año 2017 exponen el GOA [8], una técnica metaheurística que simula el comportamiento de los enjambres de saltamontes. Esta metaheurística de tipo poblacional, emplea la ubicación de saltamontes virtuales para determinar las posiciones benéficas en el espacio de búsqueda. Las posiciones de los saltamontes están determinadas por tres factores: la posición previa del saltamontes en el espacio, la influencia de todo el enjambre de saltamontes, efecto que puede ser repulsivo, atractivo o nulo, según la distancia a la cual se encuentren los individuos entre sí; por último está el efecto de búsqueda de comida, el cual se asocia al lugar en que se encuentra el mejor saltamontes en la iteración actual.

Según los resultados encontrados por sus autores, el algoritmo consigue explorar zonas prometedoras y logra a la vez que los saltamontes se agrupan alrededor del óptimo global;

características que atribuyen al efecto de relación atractiva o repulsiva entre todo el enjambre. Por lo tanto el algoritmo balancea benéficamente explotación y exploración.

Los resultados mostraron que en funciones unimodales, la tasa de atracción entre saltamontes era alta, mientras que en casos multimodales la cantidad de repulsiones entre individuos era elevada; de donde coligieron que el algoritmo evita quedar atrapado en óptimos locales, sin perjudicar su convergencia. En cuanto a las comparaciones con otras metaheurísticas, se encontró que GOA es superior a PSO [32] y FA [41] en la mayoría de problemas considerados, según el “Wilcoxon signed-rank test” [52].

1.6 Características principales de las metaheurísticas

De los anteriores párrafos se puede observar, que los algoritmos metaheurísticos poseen por lo general los siguientes rasgos distintivos: el tipo de fenómeno en que están inspirados, el uso de una población de vectores como puntos de búsqueda en el espacio Ω , su habilidad de poseer memoria o no; es decir si almacenan o no datos de iteraciones anteriores y por ultimo su habilidad de exploración y explotación eficiente del espacio de búsqueda.

Con base en esta colección de propiedades y descripciones de técnicas existentes, se abarcan claramente los fundamentos esenciales de las metaheurísticas; finalizando así el marco teórico, el cual se utiliza como punto de partida y cimiento para el diseño y descripción de la técnica metaheurística propuesta en este documento.

2. Metaheurística Diseñada

Debido a que la técnica metaheurística propuesta, se inspira en algunas características de los octópodos; como lo son: su locomoción, su anatomía, el movimiento particular de sus extremidades para realizar tareas específicas y en su capacidad cognitiva; este capítulo inicia con una descripción general del animal. Posteriormente se enuncian los rasgos biológicos apropiados y emulados en los procesos de la metaheurística, a su vez realizando una descripción de la implementación de estos en el algoritmo; seguidamente se presenta la técnica creada, describiendo su pseudocódigo, diagrama de flujo y demás elementos necesarios para su correcta interpretación.

2.1 Descripción general de los pulpos

Lo pulpos Figura 2.1, se clasifican como cefalópodos del orden de los octópodos. Son animales que desde temprana edad, al salir del huevo, subsisten eficientemente de la cacería [54], [55]. Son animales principalmente solitarios, los cuales tienen cortos periodos de vida; en algunos casos no mayores a seis meses [55].

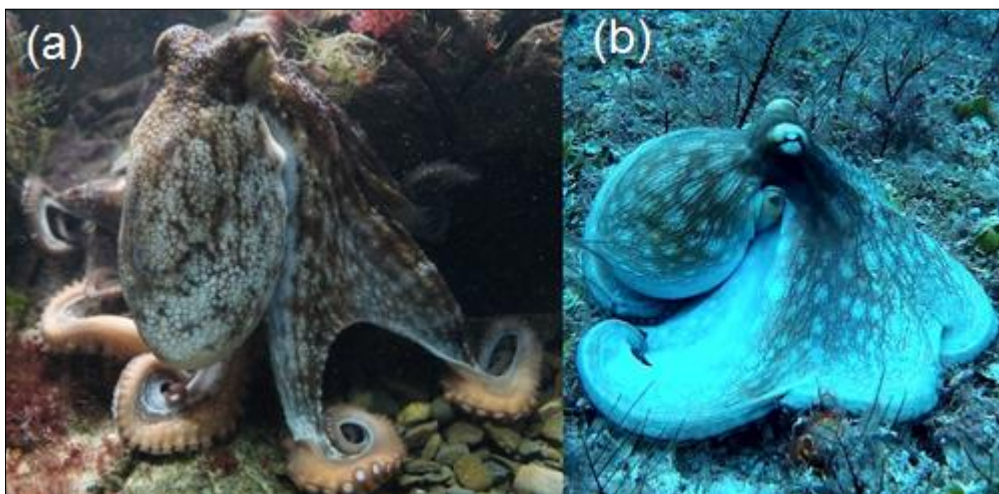


Figura 2.1 (a) Fotografía de la especie *Octopus Vulgaris*. (b) Fotografía Pulpo Arrastrándose. Fotos: Pixabay.

Físicamente se caracterizan por ser moluscos invertebrados de cuerpo blando, excepto por su pico, lo que les permite atravesar fácilmente agujeros y grietas de poca envergadura [56], [57]. Poseen ocho brazos que utilizan para desplazarse, ya sea nadando, caminando o arrastrándose; asimismo los emplean para alcanzar objetos, defenderse, explorar su entorno y recolectar comida [58]. Su piel presenta patrones de coloración que son manifestados cuando están camuflándose y varían según la posición física que el animal adopte [59], [60].

A nivel cognitivo este molusco es uno de los invertebrados más inteligentes; ya que posee personalidad, memoria espacial, consciencia de su posición y un cerebro facultado para almacenar y procesar información [61], [62]. Estas capacidades le permiten aprender de su entorno, ya sea por imitación, al observar a otros pulpos previamente entrenados o por experiencia propia [63]. En ambos casos el aprendizaje es dependiente de sus sentidos visuales y táctiles [64].

2.2 Características y movimientos emulados de los pulpos en la técnica metaheurística creada

De los anteriores atributos descritos del animal, algunos fueron utilizados como fuente de inspiración para la técnica metaheurística creada; estos rasgos fueron seleccionados según dos criterios. La influencia en la calidad de las respuestas obtenidas por el algoritmo, gracias al operador que emula esta característica y según la complejidad de ese operador en el algoritmo.

Por lo tanto cuando el operador establecido a partir de alguna propiedad del animal, no mejoraba la precisión de las soluciones o hacía que el algoritmo fuera poco esbelto; se replanteaban los procesos del operador. En caso que los resultados obtenidos gracias a este operador modificado no fueran satisfactorios; se optaba por no utilizar ese atributo como fuente de inspiración.

Durante este proceso, se realizaron modelos con atributos como la caminata, el nado, el camuflaje del animal, entre otros, los cuales fueron descartados en el algoritmo final. A continuación se exponen los rasgos biológicos utilizados en la técnica creada, explicando a la vez cómo se realiza la emulación de estos en el algoritmo.

2.2.1 Distribución espacial del animal

Físicamente este animal posee ocho brazos, los cuales están distribuidos alrededor de su cuerpo; estos se numeran del 1 al 4, precedidos por una letra R o L en el caso de ser derechos o izquierdos [65], como se ilustra en la Figura 2.2. Estos brazos poseen cientos de ventosas de una o dos filas por brazo; que se encuentran orientadas a lo largo de las extremidades y radialmente distribuidas alrededor del pico del animal [66], como se observa en la Figura 2.3.

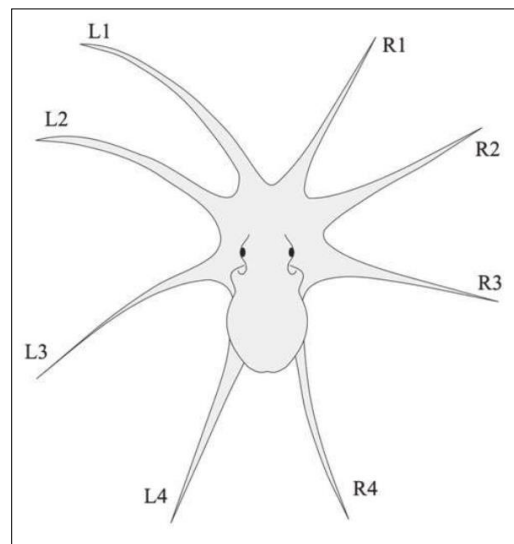


Figura 2.2 Designación de los brazos del pulpo [65].

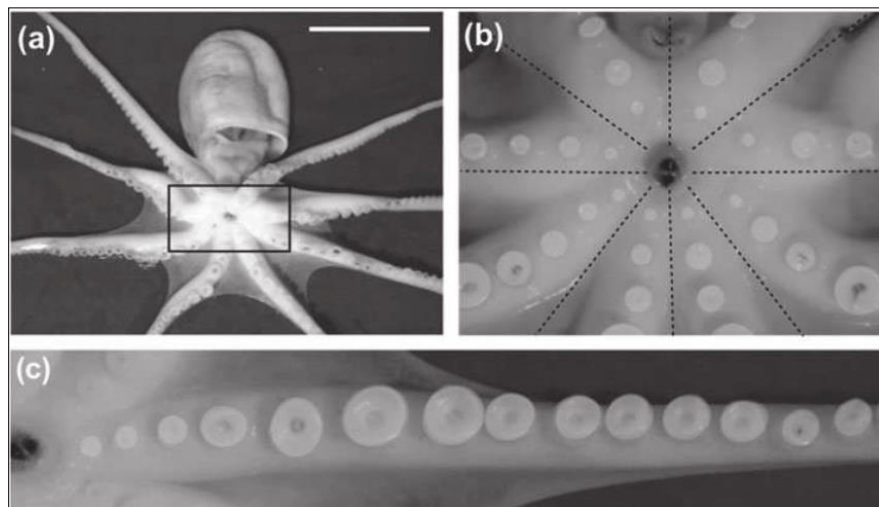


Figura 2.3 Arreglo general de ventosas de una hilera. (a) Vista del arreglo de ventosas en *E. moschata*. (b) Distribución de las ventosas alrededor del pico (recuadro negro en a). (c) Subdivisión de las ventosas en un solo brazo [67].

Para la designación de cada ventosa, se utiliza el código de identificación propuesto por Tramacere y otros [67]. Este sistema de denominación, considera el pico del animal como un punto fijo de referencia; al ser este un elemento rígido y fácilmente reconocible. Para diferenciar cada ventosa se utilizan dos rótulos: primero la denominación del brazo al cual pertenece y segundo su localización exacta en la extremidad; la cual se determina al etiquetar cada ventosa con un número ordinal, iniciando con el número uno (1) para la más cercana al pico e incrementando el valor en uno (1) por cada ventosa, hasta llegar al final de la extremidad. Por ejemplo la segunda ventosa del tercer brazo izquierdo sería L3-2.

La distribución espacial del cefalópodo en su ambiente natural, es emulada en el algoritmo por medio de la elaboración de un octópodo virtual en el espacio de búsqueda Ω . Labor que se realiza asignando posiciones coordinadas en el espacio virtual, a las ventosas $\widehat{\mathbf{S}}$ y pico $\widehat{\mathbf{B}}$ del animal; como se indica en las ecuaciones (2.1) y (2.2). Donde D equivale a las dimensiones del espacio de búsqueda y ns es el número de ventosas que hay en cada uno de los 8 brazos del animal.

$$\widehat{\mathbf{B}} = (x_1^0, x_2^0, \dots, x_D^0) \quad (2.1)$$

$$\widehat{\mathbf{S}}_{k,j} = (x_1^{k,j}, x_2^{k,j}, \dots, x_D^{k,j}); (k)_{k=1}^8; (j)_{j=1}^{ns} \quad (2.2)$$

Para representar fielmente al espécimen, las secciones de brazo $\widehat{\mathbf{T}}$ entre las ventosas del cefalópodo, son determinadas por vectores con dirección y magnitud, como se ilustra en la Figura 2.4; donde se observa el pulpo virtual para un problema de optimización de dimensión $D = 2$; adicionalmente se incluye la simbología adoptada para cada ventosa y en algunos casos su posición en el espacio de búsqueda. No se utiliza fielmente la designación propuesta por Tramacere y otros [67]; ya que en espacios multidimensionales, no es práctico emplear denominaciones definidas por la posición derecha e izquierda. Se opta por el manejo de un pulpo virtual en \mathbb{R}^n , para no restringir la aplicabilidad del algoritmo a una versión análoga fiel en \mathbb{R}^3 ; la cual únicamente permitiría 3 variables por problema.

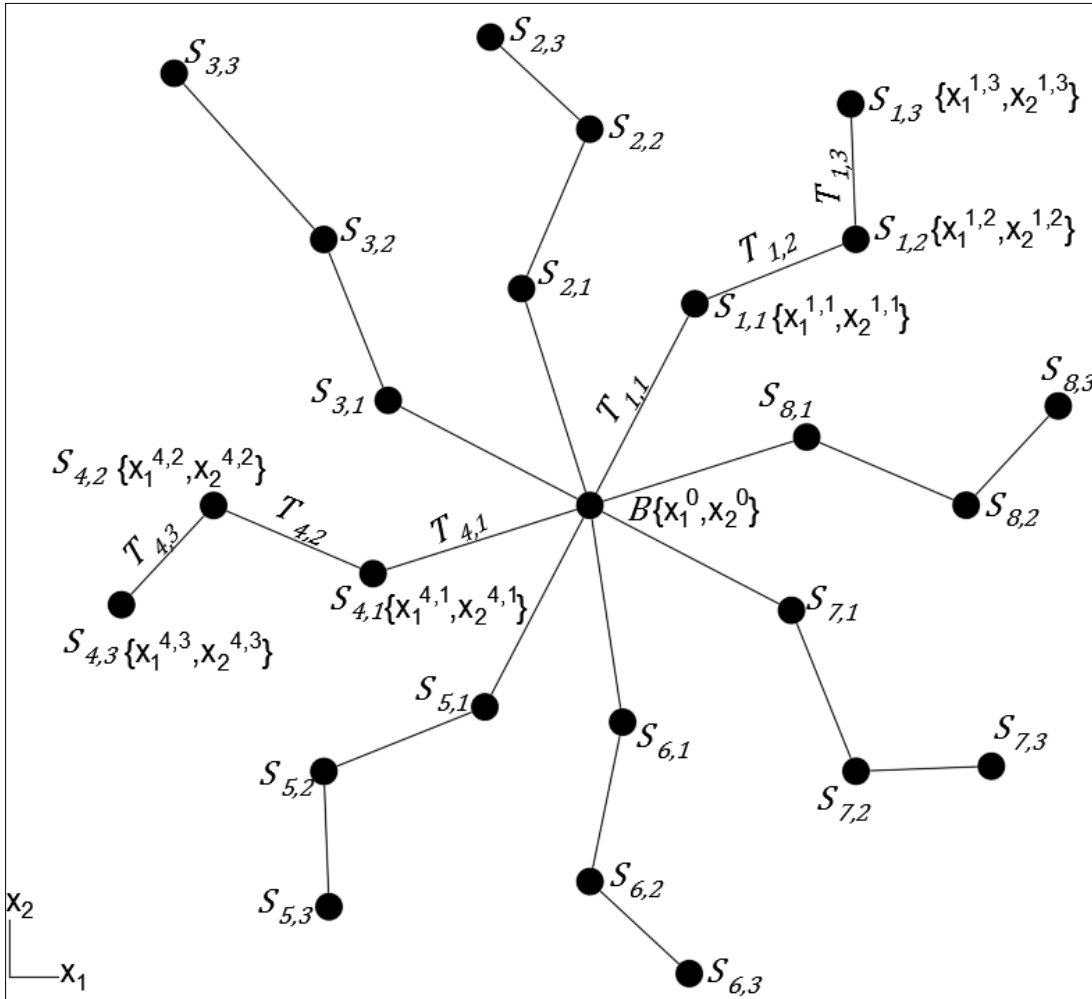


Figura 2.4 Esquema de representación del pulpo virtual en el espacio de búsqueda Ω ; caso específico para un problema de optimización con dos dimensiones y tres ventosas por brazo.

Cada una de las coordenadas (x_1, x_2, \dots, x_n) que determinan las posiciones de las ventosas y pico del animal virtual, son empleadas como variables de decisión en la función objetivo; es decir, cada una de estas ubicaciones representa una solución al problema de optimización, donde n corresponde a la dimensión D del problema de optimización. Por lo tanto se define al pico y ventosas del animal como la población de esta metaheurística; cuyo total se calcula como se indica en la ecuación (2.3).

$$N_{pop} = ns \cdot 8 + 1 \quad (2.3)$$

2.2.2 Habilidades cognitivas del pulpo

Este cefalópodo posee un sistema nervioso que compromete alrededor de 500 millones de neuronas; por lo cual es catalogado como uno de los moluscos mejor facultados cerebralmente [68]. Esta capacidad le permite tener memoria de corto y largo plazo [69]; donde guarda información adquirida de su entorno, mediante el reconocimiento visual y el aprendizaje espacial; asimismo es consciente de su posición en el espacio [61], [64], [70].

Como analogía a la memoria y conciencia espacial del animal, se establece un proceso que identifica y guarda las posiciones de interés encontradas por el octópodo virtual en el espacio de búsqueda; estas ubicaciones son definidas a continuación:

\widehat{Bp} : La mejor posición conocida por el pulpo; es decir, la ubicación que posee la mejor adecuación a la función objetivo, a través del tiempo. Es determinada según la ecuación (2.4); donde It_{max} , es el máximo número de iteraciones en la metaheurística.

$$\widehat{Bp} = Best(f(\widehat{S}_{k,j}^t)); (k)_{k=1}^8; (j)_{j=1}^{ns}; (t)_{t=1}^{It_{max}} \quad (2.4)$$

\widehat{Bs} : La mejor posición entre todas las ventosas del pulpo en la iteración actual. Para establecer esta ubicación, se excluye la posición del pico del pulpo; por lo tanto no es necesariamente la mejor posición de todo el pulpo en la iteración actual. Es determinada por la ecuación (2.5).

$$\widehat{Bs} = Best(f(\widehat{S}_{k,j}^t)); (k)_{k=1}^8; (j)_{j=1}^{ns} \quad (2.5)$$

\widehat{Bsa} : La mejor ventosa de cada brazo del octópodo. Sus valores pueden variar durante una misma iteración a medida que j varía, Figura 2.5. Está definida según lo establece la ecuación (2.6), donde se observa que \widehat{Bsa} únicamente cambia su valor si la siguiente posición encontrada en el mismo brazo es mejor a la anterior; esto se debe a que se emplea selección ávida para crear un sesgo hacia la respuesta en el algoritmo.

$$\widehat{Bsa}_{k,j} = \begin{cases} \widehat{B}^t & \text{si } f(\widehat{S}_{k,j}^t) > f(\widehat{B}^t) \quad \& \quad j = 1 \\ \widehat{S}_{k,j}^t & \text{si } f(\widehat{S}_{k,j}^t) \leq f(\widehat{B}^t) \quad \& \quad j = 1 \\ \widehat{Bsa}_{k,j-1} & \text{si } f(\widehat{S}_{k,j}^t) > f(\widehat{Bsa}_{k,j-1}) \quad \& \quad j > 1 \\ \widehat{S}_{k,j}^t & \text{si } f(\widehat{S}_{k,j}^t) \leq f(\widehat{Bsa}_{k,j-1}) \quad \& \quad j > 1 \end{cases} ; (k)_{k=1}^8; (j)_{j=1}^{ns} \quad (2.6)$$

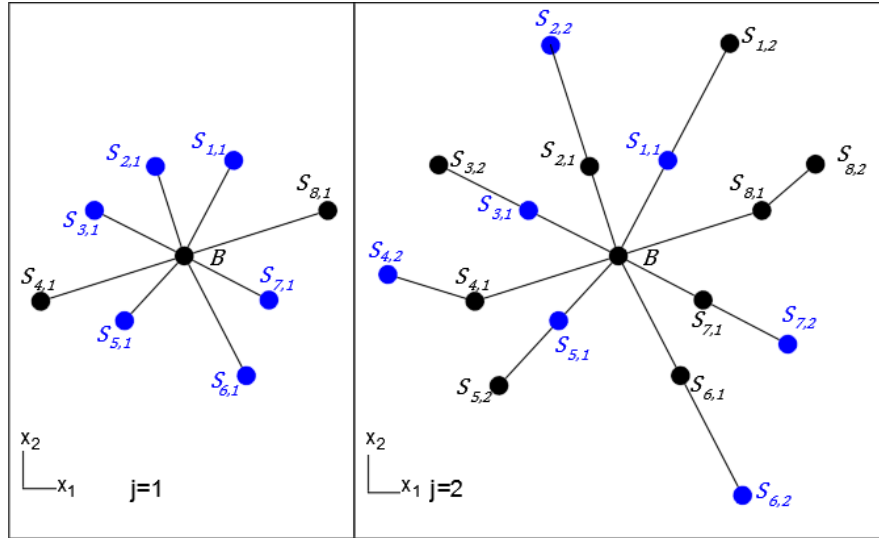


Figura 2.5 Variación de la posición de la mejor ventosa por brazo (color azul); a medida que el indicador j varia. Izquierda caso para $j=1$; tentáculos 4 y 8 poseen un valor $\widehat{Bsa} = \widehat{B}^t$. Derecha caso para $j=2$; el tentáculo 8 aún posee un valor $\widehat{Bsa} = \widehat{B}^t$.

2.2.3 Dimensión longitudinal de las extremidades del animal

Los brazos de los pulpos, catalogados como estructuras musculares hidrostáticas, se caracterizan por ser extremidades blandas carentes de sistema esquelético; capaces de producir movimientos de elongación, encogimiento, torsión y doblez [71], [72]. Según estudios realizados a la especie *Octopus Vulgaris*, las elongaciones, son en promedio del 194% [73] y en algunos casos son superiores al doble de la longitud normal del brazo [58]; existe además variabilidad en la longitud según el miembro empleado y preferencia por utilizar algunas extremidades [58], [74].

Esta variabilidad longitudinal en los brazos del animal, es emulada en la técnica metaheurística al hacer que cada sección de brazo este determinada por: el factor de elongación (ef), el factor de encogimiento (sf) y el paso entre ventosas (\widehat{L}); los cuales ocasionan fluctuación a la dimensión longitudinal de los tentáculos. La orientación de cada segmento de brazo es controlada por el vector unitario (\widehat{O}); como se puede observar en la ecuación (2.7), la cual determina como se establecen las secciones de brazo \widehat{T} .

$$\widehat{T}_{k,j} = ef_{k,j} \cdot sf^t \cdot \widehat{L}_{k,j} \cdot \widehat{O}_{k,j}; (k)_{k=1}^8; (j)_{j=1}^{ns} \quad (2.7)$$

Donde los cuatro factores que componen la ecuación anterior son:

ef : Factor de alargamiento de cada segmento de brazo, aumenta la dimensión de cada sección de tentáculo; según convenga. Su valor es igual a uno (1), a menos que se haya encontrado una mejor posición \widehat{Bsa} en la ventosa anterior del mismo brazo; es definido por la ecuación (2.8). Donde Pr , es un parámetro seleccionado por el usuario, para controlar la proporción de exploración y explotación en el algoritmo.

$$ef_{k,j} = \begin{cases} \frac{1}{10 \cdot Pr} + rand & \text{si } \begin{cases} j = 2 \ \& \ \widehat{Bsa}_{k,j-1} \neq \widehat{B}^t \\ j > 2 \ \& \ \widehat{Bsa}_{k,j-2} \neq \widehat{Bsa}_{k,j-1} \end{cases} \end{cases} \quad (k)_{k=1}^8; (j)_{j=1}^{ns} \quad (2.8)$$

sf : Factor de encogimiento de tentáculos, es un valor que disminuye la longitud del tentáculo virtual, está definido acorde a la ecuación (2.9). Su función es disminuir el área de acción del octópodo virtual a medida que transcurren las iteraciones; con el fin de intensificar la búsqueda en zonas prometedoras del espacio de búsqueda Ω .

$$sf^t = e^{\left(\left(\frac{-6}{-It_{max}}\right)^{(t-1)}\right)}; (t)_{t=1}^{It_{max}} \quad (2.9)$$

\widehat{L} : Paso entre ventosas, sus valores varían en todas las secciones de brazo y difieren para cada dimensión D ; es determinado inicialmente según el dominio del problema y posteriormente es actualizado, sin intervención directa del usuario, según se observa en la ecuación (2.10). Donde $\widehat{Y}_{j,k}$ está conformado por valores aleatorios provenientes del intervalo (0,1); $\widehat{L}t$ y $\widehat{L}s$ son los límites inferior y superior de las variables del problema de optimización y b_1 y b_2 son índices aleatorios tales que $b_1 \in \{1,2, \dots, 8\}$ y $b_2 \in \{1,2, \dots, ns\}$.

$$\widehat{L}_{k,j} = \begin{cases} \widehat{Y}_{k,j} \cdot \frac{abs(\widehat{L}t - \widehat{L}s)}{ns \cdot 8} & \text{si } t = 1 \\ \widehat{B}^{t-1} - \widehat{S}_{b_1, b_2}^{t-1} & \text{si } t > 1 \end{cases} \quad ; (k)_{k=1}^8; (j)_{j=1}^{ns} \quad (2.10)$$

\widehat{O} : Orientación de la sección de brazo en el espacio; esta es establecida por un vector unitario obtenido a partir de valores aleatorios en el intervalo (0,1). El valor de la orientación cambia para cada sección de brazo, a menos que se haya mejorado la posición \widehat{Bsa} en la ventosa anterior de ese mismo brazo; en tal caso esa dirección se reutiliza; como se expone en la ecuación (2.11).

$$\widehat{O}_{k,j} = \begin{cases} \widehat{\xi}/|\widehat{\xi}| & \text{si } j = 1 \ \text{o} \ ef = 1 \\ \widehat{O}_{k,j} = \widehat{O}_{k,j-1} & \text{si } ef \neq 1 \end{cases} \quad ; (k)_{k=1}^8; (j)_{j=1}^{ns} \quad (2.11)$$

2.2.4 Movimiento de extremidades para alcanzar objetos

El octópodo utiliza sus extremidades para alcanzar elementos de su interés; como se ve en la Figura 2.6. Durante este movimiento, el pulpo emplea uno o dos miembros a la vez, con el fin de tomar objetivos a su vista [75], [76], [77]. Este movimiento inicia usualmente con la formación de un doblado en la base del brazo a utilizar; sin embargo, es posible que se forme en cualquier lugar a lo largo del tentáculo [78]. Luego por medio de una contracción muscular, se traslada el doblado hacia la punta del brazo, a la vez orientando el miembro de manera relativamente derecha [75]; para finalmente asir el objetivo con sus ventosas.

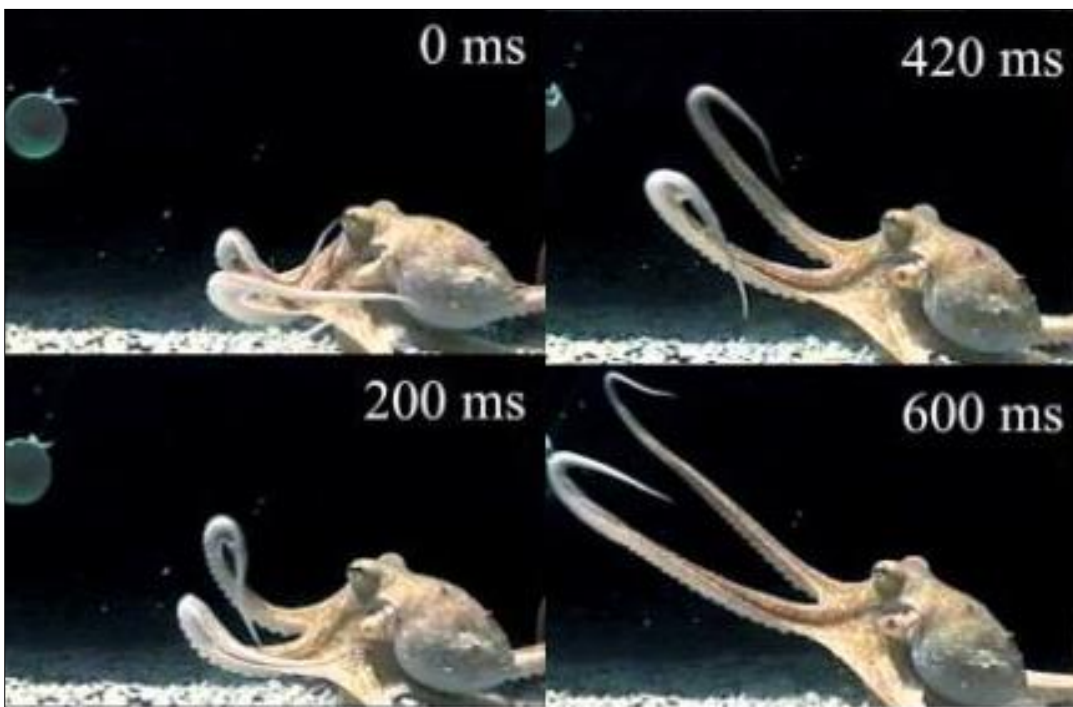


Figura 2.6 Secuencia de video que muestra 4 imágenes de un pulpo alcanzando un pequeño disco plástico con 2 brazos [75].

Este movimiento es emulado en la técnica metaheurística, al orientar algunos segmentos de cada tentáculo hacia posibles objetivos, Figura 2.7. Como no se conoce el espacio de búsqueda a priori, se supone que tales objetivos se encuentran en direcciones prometedoras, a lo largo de las cuales se ubicarán las secciones de los brazos. Las direcciones son definidas como prometedoras, si al haber sido empleadas para hallar la ventosa anterior de ese brazo, permitieron mejorar la posición de B_{sa} .

Cuando una dirección es definida como prometedora, es almacenada y reutilizada para la siguiente ventosa de ese brazo, según se definió en la ecuación (2.11). Adicionalmente en el segmento de brazo orientado hacia un rumbo prometedor, se utiliza el factor de extensión ef , ecuación (2.8); con el propósito de aumentar el alcance en esa trayectoria.

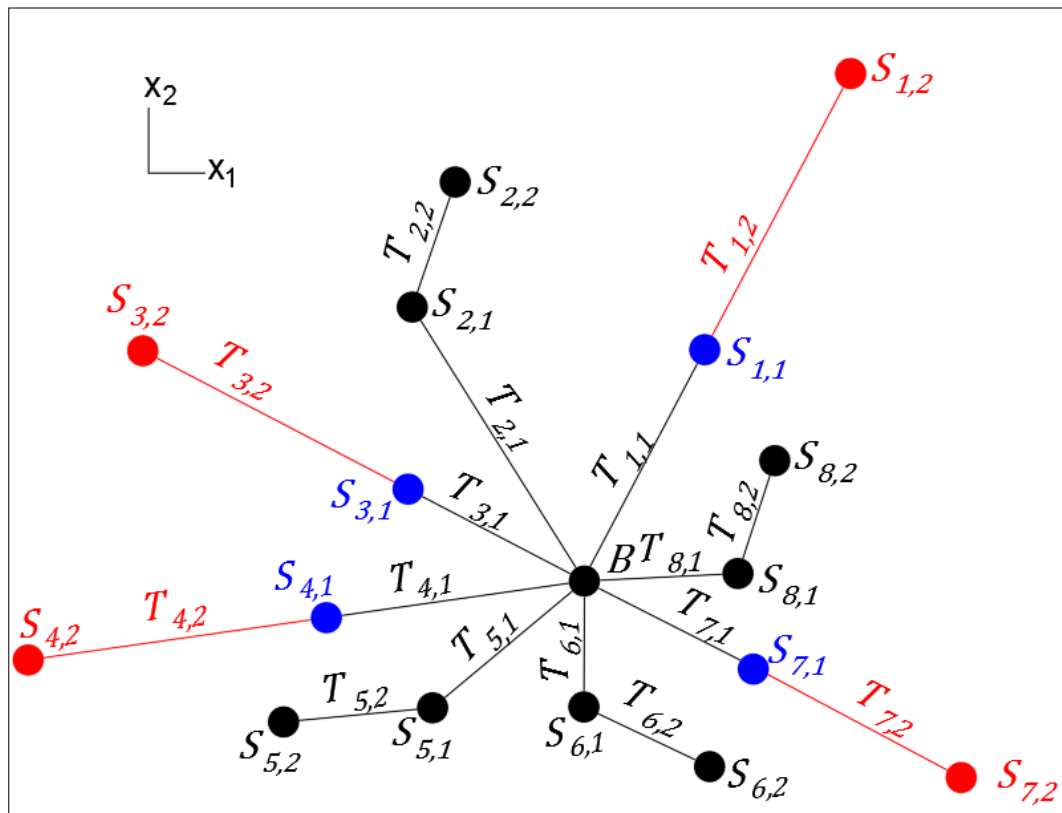


Figura 2.7 Grafico en \mathbb{R}^2 que representa la emulación de la orientación de extremidades del pulpo; en rojo se observan los segmentos que sufrieron orientación y elongación, en azul las ventosas \widehat{Bsa} .

2.2.5 Regeneración de miembros perdidos

En un ambiente como el mar, lleno de predadores y otras clases de peligros, los pulpos son susceptibles a sufrir pérdida de extremidades, ya sea física o funcionalmente; el animal puede hacer frente a esta mutilación o posible atrofia, mediante un proceso de renovación, crecimiento y restauración de tejidos y órganos deteriorados [79], [80].

Esta característica regenerativa fue minuciosamente estudiada por Fossati y otros [81]; allí se analizó la influencia de la enzima Acetilcolinesterasa (AChE) en la regeneración de los

brazos del animal. Para ello se recortaron porciones de uno y dos centímetros de largo en extremidades de pulpos pertenecientes a la especie *Octopus Vulgaris*, siendo las mutilaciones del orden del 4% de la longitud normal de los brazos. Cabe resaltar que en un periodo de 55 días, el miembro del animal mostro una estructura completa que ostentaba una restauración de todos los componentes faltantes [81], Figura 2.8; verificando así el extraordinario potencial regenerativo que poseen estos cefalópodos.

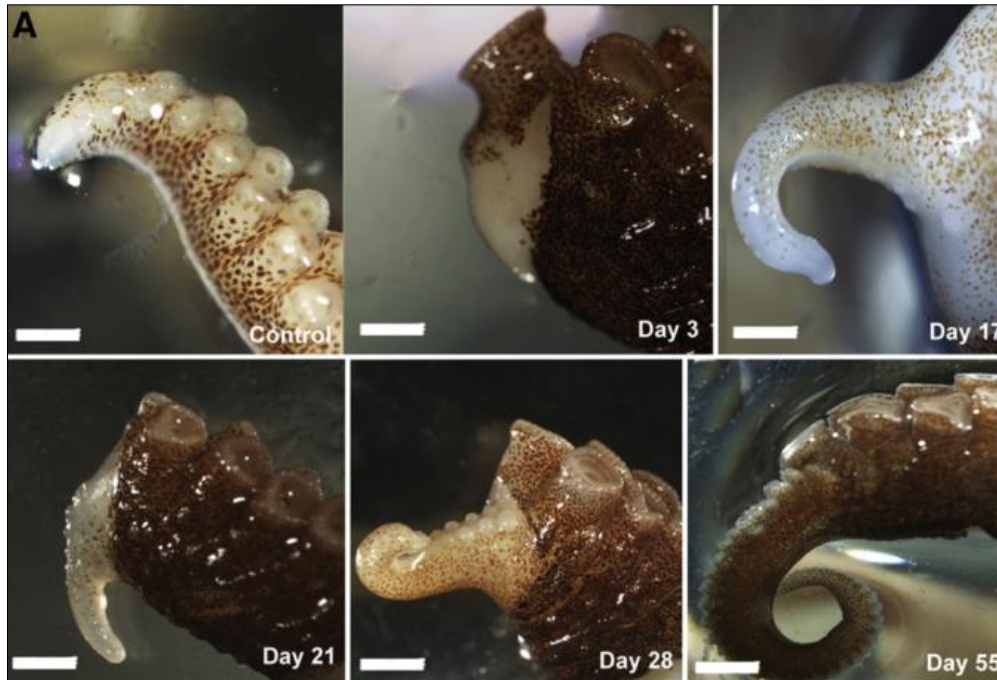


Figura 2.8 Imágenes de la evolución de la extremidad mutilada. En el recuadro A, se observa el animal de control; en los otros recuadros se muestra la evolución del brazo mutilado desde el día 3 al 55 [81].

La técnica metaheurística emula la propiedad regenerativa en los brazos de este octópodo, como dos procesos que persiguen el mismo fin; el cual es la obtención de nuevas posiciones aleatorias, para las ventosas virtuales en el espacio de búsqueda. El primer operador (regeneración 1) define que durante la ejecución del algoritmo, el cefalópodo virtual puede según una probabilidad P_r , sufrir una pérdida parcial o deterioro de sus extremidades. Posteriormente estos miembros al regenerarse, se ubican dentro del dominio del problema, en una posición distinta a la inicial; proceso observado en la Figura 2.9 y descrito por la ecuación (2.12).

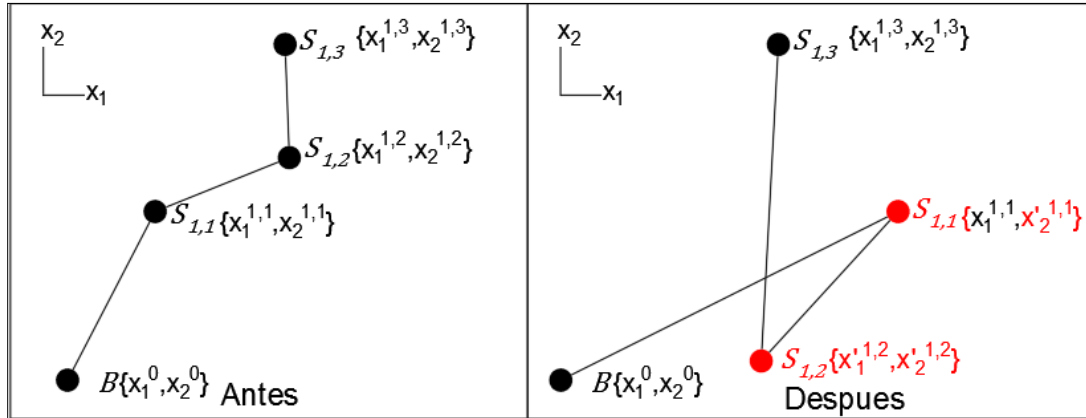


Figura 2.9 Gráfico en \mathbb{R}^2 que representa la acción del operador regeneración 1. Se muestra únicamente uno de los brazos afectados; donde las ventosas \hat{S}_{11} y \hat{S}_{12} sufrieron cambio.

$$\hat{S}_{i,k,j} = \begin{cases} \hat{S}_{i,k,j} & \text{si } rand_1 > P_r \\ \widehat{L}\bar{l}_i + abs(\widehat{L}\bar{l}_i - \widehat{L}\bar{s}_i) \cdot rand_2 & \text{si } rand_1 \leq P_r \end{cases}; (k)_{k=1}^B; (j)_{j=1}^{ns}; (i)_{i=1}^D \quad (2.12)$$

El segundo operador (regeneración 2) es ejecutado siempre una vez por iteración. Durante este proceso, se selecciona aleatoriamente una ventosa al final de alguna extremidad, la cual sufre un reemplazo total de las coordenadas que la conforman; es decir, su nueva posición se establece aleatoriamente dentro del espacio de búsqueda. Según se observa en la Figura 2.10 y en la ecuación (2.13)

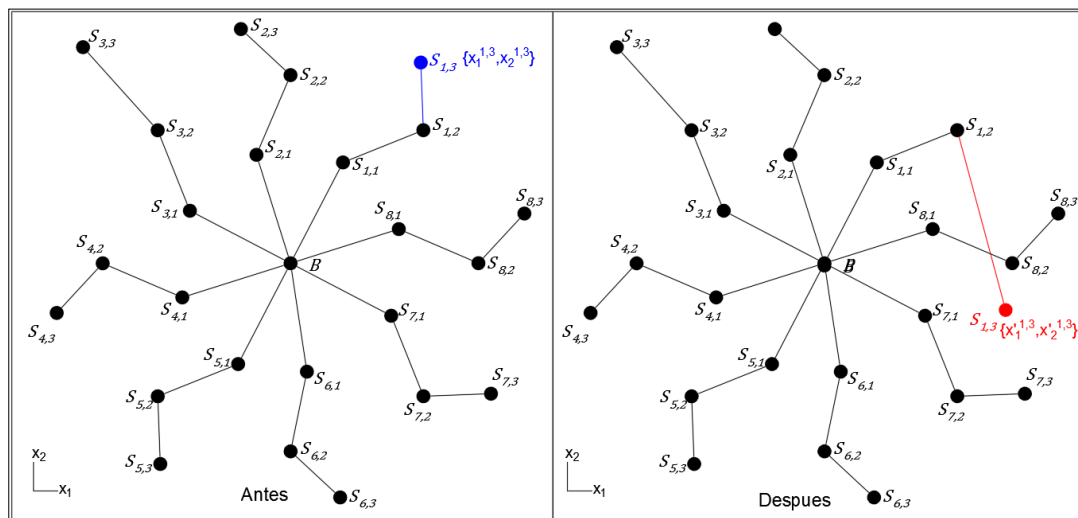


Figura 2.10 Grafico en \mathbb{R}^2 que representa la acción del operador regeneración 2 en un brazo seleccionado aleatoriamente; donde la ventosa \hat{S}_{13} se cambió en su totalidad.

$$\hat{S}_{b_4,ns} = \hat{L}i + \text{abs}(\hat{L}i - \hat{L}s) \cdot \hat{\zeta} \quad (2.13)$$

Donde $\hat{\zeta}$ es un vector de tamaño $D_x \cdot 1$ con valores aleatorios provenientes del intervalo (0,1); $\hat{L}i$ y $\hat{L}s$ son los límites inferior y superior de las variables del problema de optimización, b_4 es un índice aleatorio perteneciente al conjunto {1,2, ...,8}.

Estos dos operadores difieren entre sí, el proceso regeneración 2 hace que el cambio ocurra sin excepción todas las iteraciones, además modifica todas las coordenadas de posición en una ventosa aleatoriamente seleccionada; en cambio en el proceso regeneración 1, la modificación de la ventosa depende de la probabilidad Pr y la alteración ocurre generalmente solo en unas de sus coordenadas.

2.2.6 Movimientos para el desplazamiento (Locomoción)

El pulpo emplea sus extremidades para llevar a cabo tareas motrices como lo son: la locomoción, la recolección de alimentos, la cacería y también la manipulación compleja de objetos [75]. Labores que puede ejecutar con una amplia gama de movimientos [76].

Dentro del área de la locomoción se encuentran cuatro variantes identificadas: el nado, la caminata bípeda, la propulsión a chorro y el arrastramiento; patrones de movimiento utilizados por el animal para desplazarse de un lugar a otro [82]. De las anteriores cuatro, únicamente se emulan en el algoritmo, el arrastramiento Figura 2.11 (a) y la propulsión a chorro Figura 2.11 (b); formas de traslación que entre sí, difieren bastante con respecto a la ubicación de los brazos en el espacio.

La propulsión a chorro Figura 2.11 (b), es una forma de nado veloz donde el cefalópodo, ingresa agua en la cavidad del manto, para luego expulsarla rápidamente por un ducto tubular denominado hiponome; logrando así impulsarse en sentido contrario a la eyección del líquido, tal como sucede en una propulsión a chorro [83], [84]. Durante este movimiento los brazos se encuentran derechos y fuertemente juntos tras del manto, en una posición donde el cuerpo sigue una trayectoria rectilínea [82]; por lo tanto ningún brazo presenta movimientos de exploración y no hay ventosas escudriñando el espacio [70].

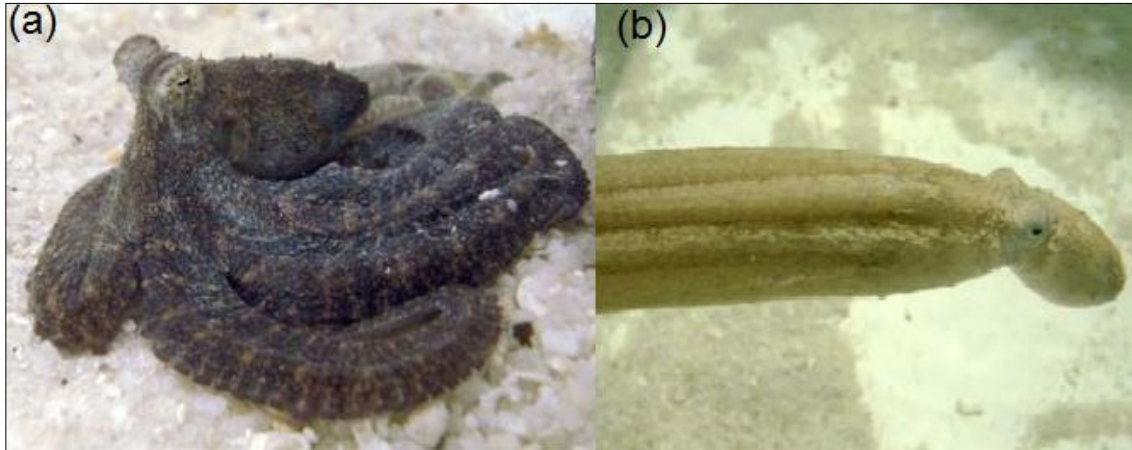


Figura 2.11 Locomoción del pulpo *Abdopus aculeatus*. (a) Arrastramiento; (b) Propulsión a chorro [82].

La propulsión a chorro es empleada por los pulpos para viajar rápidamente, como se ha observado en las especies *Octopus Vulgaris* [85] y *Octopus Cyanea Grey* [86]. Las cuales usan este movimiento, como medio de escape cuando sus habilidades de defensa fallan; adicionalmente lo utilizan al alejarse de su morada, mientras encuentran una localización propicia para dar inicio a sus sesiones de cacería [84], [87].

La implementación de esta maniobra en la metaheurística, se realiza con un operador que considera únicamente al pico virtual del animal; ubicándolo en una posición aleatoria en el espacio de búsqueda. Este operador se utiliza para definir en la primera iteración del algoritmo, la posición inicial del pico del pulpo, de acuerdo a la ecuación (2.14). También se emplea durante las primeras iteraciones del algoritmo, para posiblemente reiniciar la posición del pico del pulpo en la siguiente iteración; según se observa en la ecuación (2.15); con esto se pretende diversificar las soluciones iniciales del algoritmo y evitar una convergencia prematura.

$$\hat{\mathbf{B}}^t = \hat{\mathbf{L}}\mathbf{i} + \text{abs}(\hat{\mathbf{L}}\mathbf{i} - \hat{\mathbf{L}}\mathbf{s}) \cdot \hat{\zeta} \quad (2.14)$$

$$\hat{\mathbf{B}}^{t+1} = \hat{\mathbf{L}}\mathbf{i} + \text{abs}(\hat{\mathbf{L}}\mathbf{i} - \hat{\mathbf{L}}\mathbf{s}) \cdot \hat{\zeta} \quad \text{si } \text{rand} \leq P_r \quad \& \quad t \leq \lfloor It_{max} \cdot 10 \cdot P_r \rfloor \quad (2.15)$$

El otro método de traslación emulado es el arrastramiento, Figura 2.11 (a). Este consiste en un desplazamiento a lo largo del fondo del mar, en el cual el cefalópodo utiliza sus brazos para halarse o empujarse; tarea que realiza con ayuda de sus ventosas, las cuales están en un proceso constante de adherencia y liberación con el sustrato del mar [86], [87], [88].

El arrastramiento además de ser la forma más común de desplazamiento, como sucede en las especies *Octopus cyanea* [86] y *Abdopus aculeatus* [82], [89]; es también la forma más diversa de locomoción, la cual incorpora tantas posturas que pocas han podido ser definidas [82]. Esa versatilidad de posiciones, permitió en el algoritmo definir localizaciones diversas para las ventosas virtuales; logrando así examinar eficazmente el espacio de búsqueda.

Para simular esta forma de traslación, se toma inicialmente la posición del pico del cefalópodo y alrededor de este se definen las ventosas. Seguidamente se determina la posición $\widehat{\mathbf{B}}\mathbf{s}$ en la iteración actual; ya que es allí donde generalmente se ubicara el pico del pulpo en la iteración siguiente. Este procedimiento se describe detalladamente a continuación:

Inicialmente, se definen las primeras ventosas de cada brazo ($\widehat{\mathbf{S}}_{k,1}$), según la ecuación (2.16) para $j = 1$. Donde $\widehat{\mathbf{T}}$ se obtiene según la ecuación (2.7) como se indicó previamente.

$$\widehat{\mathbf{S}}_{k,j} = \begin{cases} \widehat{\mathbf{B}}^t + \widehat{\mathbf{T}}_{k,j} & \text{si } j = 1 \\ \widehat{\mathbf{B}}\mathbf{sa}_{k,j-1} + \widehat{\mathbf{T}}_{k,j} & \text{si } j > 1 \end{cases}; (k)_{k=1}^8; (j)_{j=1}^{ns} \quad (2.16)$$

Después de la ubicación de las ventosas más cercanas al pico ($\widehat{\mathbf{S}}_{k,1}$), se realiza la evaluación de estas en la función objetivo; para seleccionar las mejores ventosas por brazo ($\widehat{\mathbf{B}}\mathbf{sa}$), según la ecuación (2.6). Luego con base en estos resultados y utilizando la ecuación (2.16), se pueden establecer las siguientes ventosas de cada brazo $\widehat{\mathbf{S}}_{k,2}$ y realizar nuevamente el proceso previamente descrito. Este procedimiento ilustrado en la Figura 2.12, se realiza para $j = 1, 2, \dots, ns$; es decir hasta definir la totalidad de ventosas del octópodo.

Luego de haber finalizado la distribución del pulpo en el espacio, se establece la mejor posición entre todas las ventosas del pulpo en la iteración actual ($\widehat{\mathbf{B}}\mathbf{s}$) mediante la ecuación (2.5); esta posición se utiliza para trasladar allí al pico del pulpo en la iteración siguiente, conforme a la ecuación (2.17); simulando así los movimientos en los que el pulpo utiliza sus brazos y ventosas para desplazar su cuerpo, hacia una dirección específicamente seleccionada.

Esto ocurre a menos que el animal virtual decida viajar a una posición diferente, como puede suceder en las primeras iteraciones del algoritmo, al simular la maniobra de propulsión a chorro ecuación (2.15) o en caso que el animal decida dirigirse a la mejor posición conocida ($\widehat{\mathbf{B}}\mathbf{p}$), como se establece en la ecuación (2.17).

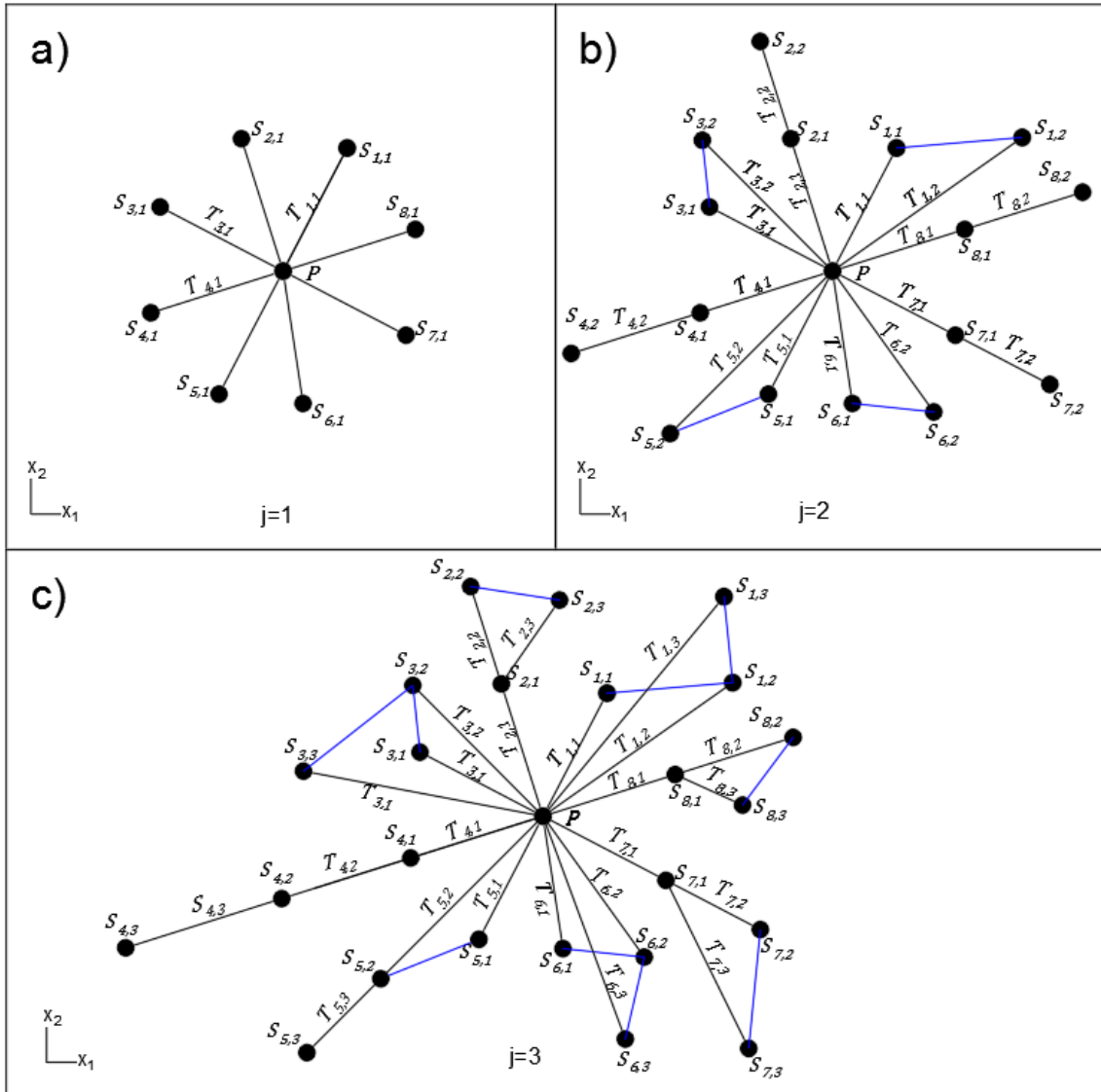


Figura 2.12 Imagen de un pulpo virtual 2D, en el cual se muestra el proceso de establecimiento de ventosas para $ns = 3$. a) Ventosas iniciales $j = 1$; b) ventosas en $j = 2$; c) ventosas para $j = ns$.

$$\widehat{\mathbf{B}}^{t+1} = \begin{cases} \widehat{\mathbf{B}}\mathbf{p} & \text{si } rand \leq 1 - P_r \quad \& \quad t > \lfloor It_{max} \cdot 10 \cdot P_r \rfloor \\ \widehat{\mathbf{B}}\mathbf{s} & \text{si } \begin{cases} rand \geq 1 - P_r \quad \& \quad t > \lfloor It_{max} \cdot 10 \cdot P_r \rfloor ; (t)_{t=1}^{It_{max}} \\ rand \geq P_r \quad \& \quad t \leq \lfloor It_{max} \cdot 10 \cdot P_r \rfloor \end{cases} \end{cases} \quad (2.17)$$

A partir de esta nueva posición $\widehat{\mathbf{B}}^t$ del animal, se distribuirán nuevamente las ventosas en la iteración siguiente; se prosigue de la misma manera hasta llegar al final de las iteraciones.

2.3 Algoritmo de optimización del pulpo artificial (Artificial Octopus Optimization Algorithm- AOOA)

En párrafos anteriores se describen de forma aislada los operadores de esta metaheurística; sin embargo para dar claridad al algoritmo diseñado, a continuación se especifica el proceso que realiza el algoritmo, enunciando ordenadamente sus operadores.

El proceso de la metaheurística inicia con la especificación del problema a optimizar; es decir, se realiza la definición de la función objetivo $f(\hat{x})$, el espacio de búsqueda ($\hat{L}l$ y $\hat{L}s$) y en caso de ser necesarias las restricciones del problema de optimización; posteriormente el usuario establece la cantidad de ventosas por brazo (ns), el valor Pr y el número máximo de iteraciones (It_{max}).

Se prosigue con la creación de los valores iniciales para el paso entre ventosas \hat{L} , según la ecuación (2.10); luego por medio del operador de propulsión, se crea la posición inicial del pico virtual del animal (\hat{B}), tarea que se realiza por medio de la ecuación (2.14).

Se prosigue con el ciclo iterativo, definiendo inicialmente el factor de encogimiento de tentáculos (sf), como se indica en la ecuación (2.9); después se obtienen las posiciones de las primeras ventosas para esa iteración ($\hat{S}_{k,1}^t$), mediante la ecuación (2.16); a continuación se realiza la regeneración de miembros perdidos, conforme a las ecuaciones (2.12) y (2.13) según corresponda; posterior a ello se realiza la selección de las mejores ventosas por brazo \widehat{Bsa} , de acuerdo a la ecuación (2.6). Este proceso de establecer ventosas, continua según el operador de arrastramiento en la página 56.

A continuación se lleva a cabo la actualización del paso entre ventosas \hat{L} , ecuación (2.10); luego se selecciona la mejor posición de las ventosas del pulpo en la iteración actual (\widehat{Bs}), ecuación (2.5) y se actualiza, en caso de ser necesario, la mejor posición conocida del octópodo (\widehat{Bp}), ecuación (2.4).

Después se define la nueva posición del pico virtual del animal, ecuaciones (2.15) y (2.17); luego de esto, se actualiza el valor de la iteración actual t , al incrementar en uno su valor, proceso que se repite hasta finalizar el ciclo iterativo. Por último se declara la mejor posición del pulpo virtual (\widehat{Bp}) y el valor mínimo obtenido para la función objetivo $f(\widehat{Bp})$.

2.3.1 Pseudocódigo de AOOA

Defina

$t = 1$

La función objetivo $f(\hat{x})$

Límites de variables y restricciones \hat{L}_i, \hat{L}_s

Número máximo de iteraciones It_{max}

Valor de relación entre exploración y explotación Pr

Ventosas por brazo ns

Inicialice

Paso entre ventosas \hat{L} , ecuación (2.10)

Pico del pulpo \hat{B} , ecuación (2.14)

Ciclo 1 while $t \leq It_{max}$

Defina el factor de encogimiento del tentáculo sf , ecuación (2.9)

Ciclo 2 for $j = 1: ns$

Defina la posición de las ventosas $\hat{S}_{k,j}^t$, ecuación (2.16)

Ejecute

Operador regeneración 1, ecuación (2.12)

Si $j = ns$

Ejecute

Operador regeneración 2, ecuación (2.13)

Fin si

Almacene los resultados de las modificaciones de los operadores en las ventosas $\hat{S}_{k,j}^t$

Actualice y almacene los valores de \hat{Bsa} , ecuación (2.6)

$j=j+1$;

Fin ciclo 2

Actualice y almacene el Paso entre ventosas \hat{L} , ecuación (2.10)

Actualice y almacene la mejor posición de las ventosas del pulpo \hat{Bs} , ecuación (2.5)

Actualice y almacene la mejor posición conocida del octópodo \hat{Bp} , ecuación (2.4)

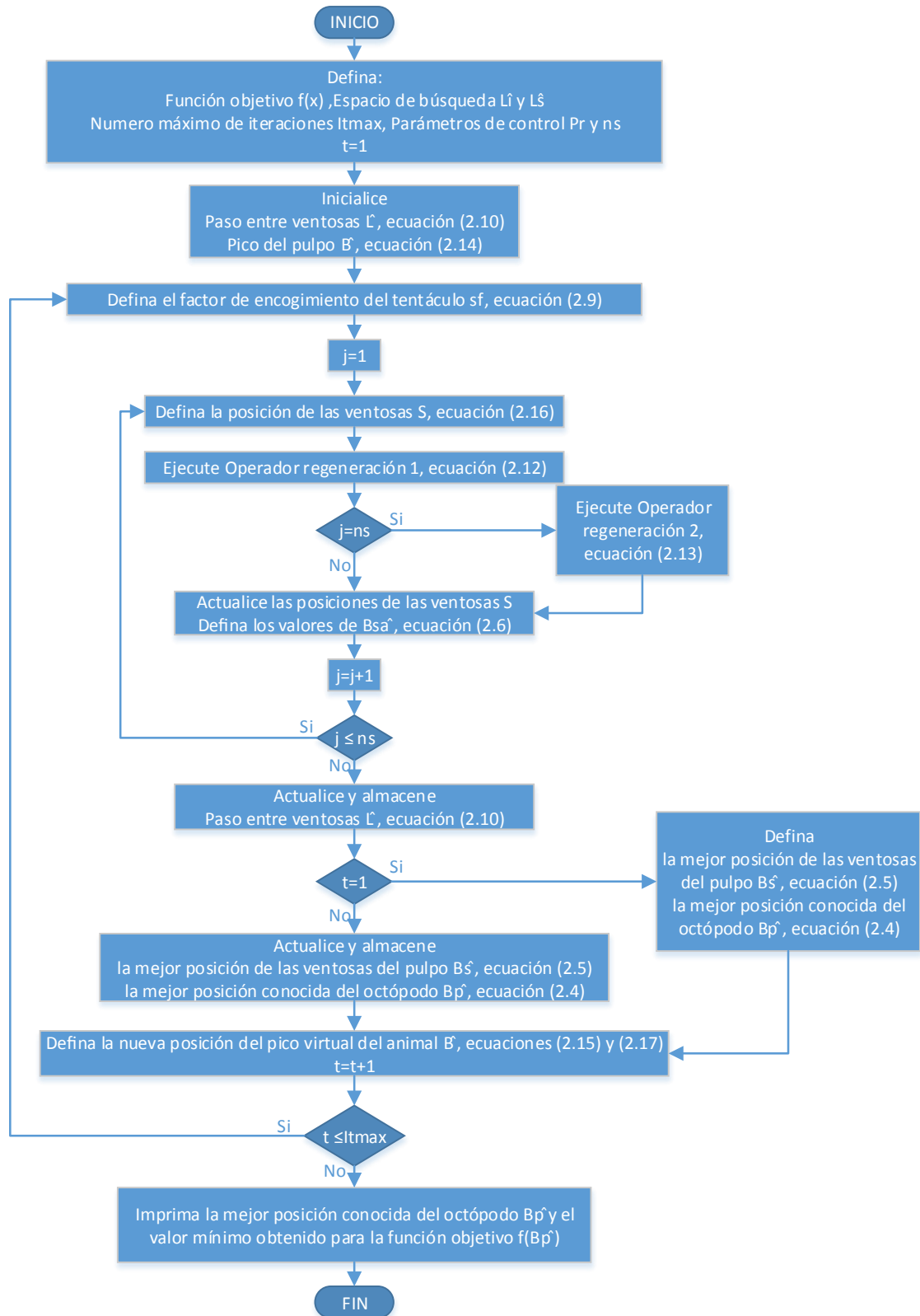
Defina la nueva posición del pico virtual del animal \hat{B} , ecuaciones (2.15) y (2.17);

$t = t + 1$

Fin ciclo 1

Imprima el valor de \hat{Bp} la mejor posición conocida del octópodo y el valor mínimo obtenido para la función objetivo $f(\hat{Bp})$.

2.3.2 Diagrama de flujo de AOOA



3. Desempeño del Algoritmo

Este capítulo inicia con la descripción de un conjunto de funciones de prueba empleadas en la evaluación de desempeño de AOOA, se prosigue con la evaluación de este conjunto utilizando la técnica creada; luego se presentan comparaciones entre AOOA y otras técnicas consideradas. Por último se realiza una explicación de la configuración de parámetros seleccionada para el algoritmo diseñado y un análisis de sensibilidad de respuesta, según la variación de sus parámetros.

3.1 Problemas de prueba utilizados para la evaluación del algoritmo

Con el fin de examinar la eficiencia y fiabilidad de las técnicas metaheurísticas en hallar el óptimo global, se emplean diferentes conjuntos de funciones de prueba [90]. Estos grupos, deben componerse preferiblemente de funciones de diferentes tipos; para así poder evaluar en un amplio espectro, la capacidad del algoritmo para buscar el óptimo global [91].

En la literatura no existe aún, un conjunto definitivo de funciones de prueba para evaluar algoritmos metaheurísticos [90]; además, diseñar un grupo ideal de problemas donde se seleccionen todos los tipos de funciones, es una tarea infructuosa [91]. Razón por la cual, se utilizará para la validación del algoritmo, un subconjunto de funciones empleado en estudios comparativos publicados [91], [92]; el cual posee las características deseadas para la evaluación de técnicas metaheurísticas.

En la Tabla 1, se presenta el conjunto de problemas seleccionado con su respectiva clasificación; especificando si los problemas son de tipo unimodal o multimodal y separable o no separable. Además se incluye el valor mínimo teórico de cada función, el dominio donde se evalúa y la cantidad de variables por problema.

Tabla 1. Conjunto de funciones empleadas para las pruebas experimentales. D: dimensión de la función, T: tipo de función (U: unimodal, M: multimodal, S: separable, N: no-separable), No: número.

No	Nombre	T	D	Dominio	Mínimo
1	Sphere	US	30	[-100, 100]	0
2	Sum Squares	US	30	[-10, 10]	0
3	Beale	UN	5	[-4.5, 4.5]	0
4	Easom	UN	2	[-100, 100]	-1
5	Matyas	UN	2	[-10, 10]	0
6	Colville	UN	4	[-10, 10]	0
7	Trid 6	UN	6	[-36, 36]	-50
8	Trid 10	UN	10	[-100, 100]	-210
9	Zakharov	UN	10	[-5, 10]	0
10	Powell	UN	24	[-4, 5]	0
11	Schwefel 2.22	UN	30	[-10,10]	0
12	Rosenbrock	UN	30	[-30, 30]	0
13	Dixon-Price	UN	30	[-10, 10]	0
14	Foxholes	MS	2	[-65.536,65.536]	0.998003
15	Branin	MS	2	[-5, 10],[0,15]	0.397887
16	Bohachevsky 1	MS	2	[-100, 100]	0
17	Booth	MS	2	[-10, 10]	0
18	Rastrigin	MS	30	[-5.12, 5.12]	0
19	Schwefel	MS	30	[-500, 500]	-12569.4866
20	Michalewicz 2	MS	2	[0, π]	-1.8013
21	Michalewicz 5	MS	5	[0, π]	-4.687658
22	Michalewicz 10	MS	10	[0, π]	-9.66015
23	Schaffer	MN	2	[-100, 100]	0
24	Six Hump Camel Back	MN	2	[-5, 5]	-1.0316
25	Bohachevsky 2	MN	2	[-100, 100]	0
26	Bohachevsky 3	MN	2	[-100, 100]	0
27	Shubert	MN	2	[-10, 10]	-186.7309
28	Goldstein-Price	MN	2	[-2, 2]	3
29	Kowalik	MN	4	[-5, 5]	0.00031
30	Shekel 5	MN	4	[0, 10]	-10.1532
31	Shekel 7	MN	4	[0, 10]	-10.4029
32	Shekel 10	MN	4	[0, 10]	-10.5364
33	Perm	MN	4	[-4, 4]	0
34	Power Sum	MN	4	[0, 4]	0
35	Hartman 3	MN	3	[0, 1]	-3.86278
36	Hartman 6	MN	6	[0, 1]	-3.32237
37	Griewank	MN	30	[-600, 600]	0
38	Ackley	MN	30	[-32, 32]	0
39	Langerman5	MN	5	[0,10]	-1.5
40	FletcherPowell2	MN	2	[- π , π]	0
41	FletcherPowell5	MN	5	[- π , π]	0
42	FletcherPowell10	MN	10	[- π , π]	0

Debido a que hasta el momento, no se conoce un algoritmo efectivo para resolver todos los tipos de funciones eficientemente [92]; se deben identificar las propiedades del algoritmo y a la vez encontrar qué tipo de funciones este resuelve adecuadamente [90], [92], [93].

Para realizar esta labor, es necesario conocer las características de cada tipo de problema, ya que según estas, se juzgan cierto tipo de habilidades en el algoritmo; por lo cual, a continuación se realiza una descripción de las propiedades de cada función y las capacidades que evalúa.

La modalidad determina si la función posee o no mínimos locales; en consecuencia, las funciones multimodales son utilizadas a fin de verificar la habilidad del algoritmo, para escapar a mínimos locales y explorar eficientemente el espacio de búsqueda [91], [92]. A medida que incrementa la dimensión de los problemas de optimización, se eleva la dificultad para resolverlos [90], [92], [93]. La separabilidad o no de la función, permite identificar si sus variables son o no dependientes entre sí; las funciones separables son usualmente más sencillas de resolver que las no separables [92].

Además de las características ya mencionadas, se consideran otras propiedades presentes en algunas de las funciones de la Tabla 1. Por ejemplo, están las funciones de superficie plana como Matyas y PowerSum, las cuales no proveen información en la dirección de búsqueda [90], [92]. Otras funciones poseen el óptimo global cerca de óptimos locales, como ocurre en las funciones Kowalik, Perm y Schaffer, las cuales exigen que el algoritmo posea estupendos operadores de búsqueda local [92], [91]. Las funciones Easom, Michalewicz 10 y Powell, consideradas de alta complejidad; poseen el óptimo global en una zona angosta y pequeña en comparación del espacio de búsqueda [90], [91], [92]. Los problemas de Goldstein-Price y Trid, tienen grado polinomial alto y problemas de escalabilidad [90], [92]. Las funciones con un valle de curvatura estrecho como Rosenbrock, Beale y Colville, hacen que sea difícil seguir los cambios de dirección de la función [90], [91].

3.2 Evaluación de desempeño con respecto a problemas de prueba

A continuación se evalúa el desempeño del algoritmo, al resolver los problemas de optimización listados en la Tabla 1. Para la obtención de estos resultados, las configuraciones de los parámetros son: $Pr = 0.0175$ y $ns = 3$; obteniendo veinticuatro (24) ventosas en la totalidad del octópodo virtual. Se realizaron 30 ejecuciones del algoritmo por función, permitiendo en cada una de ellas como máximo, cien mil (100,000) evaluaciones de la función objetivo, los resultados obtenidos se encuentran en la Tabla 2.

Tabla 2. Resultados estadísticos de evaluar las 42 funciones de prueba con AOOA, para cien mil evaluaciones de la función objetivo. D: dimensión de la función, T: tipo de función, Min.Obt: mínimo valor obtenido, Prom.: promedio de los mejores valores obtenidos en las 30 ejecuciones del algoritmo, D. Estd: desviación estándar.

No	Nombre	T	D	Dominio	Mínimo	Min. Obt.	Prom.	Desv. Estd.
1	Sphere	US	30	[-100, 100]	0	1.205E-18	9.744E-18	1.207E-17
2	Sum Squares	US	30	[-10, 10]	0	9.873E-20	7.903E-19	6.388E-19
3	Beale	UN	5	[-4.5, 4.5]	0	0	0	0
4	Easom	UN	2	[-100, 100]	-1	-1	-1	0
5	Matyas	UN	2	[-10, 10]	0	2.321E-63	4.273E-47	1.401E-46
6	Colville	UN	4	[-10, 10]	0	2.817E-06	0.015209	0.014153
7	Trid 6	UN	6	[-36, 36]	-50	-50	-50	1.660E-13
8	Trid 10	UN	10	[-100, 100]	-210	-210	-209.999999	7.243E-07
9	Zakharov	UN	10	[-5, 10]	0	6.324E-17	2.505E-15	4.358E-15
10	Powell	UN	24	[-4, 5]	0	0.002025	0.003182	0.000503
11	Schweffel 2.22	UN	30	[-10,10]	0	9.156E-11	3.395E-10	1.885E-10
12	Rosenbrock	UN	30	[-30, 30]	0	0.196337	46.199496	39.314685
13	Dixon-Price	UN	30	[-10, 10]	0	3.940E-08	0.752966	0.562284
14	Foxholes	MS	2	[-65.536,65.536]	0.998003	0.998003	0.998003	1.090E-16
15	Branin	MS	2	[-5, 10],[0,15]	0.397887	0.397887	0.397887	0
16	Bohachevsky 1	MS	2	[-100, 100]	0	0	0	0
17	Booth	MS	2	[-10, 10]	0	0	0	0
18	Rastrigin	MS	30	[-5.12, 5.12]	0	0	1.293E-10	7.063E-10
19	Schweffel	MS	30	[-500, 500]	-12569.4866	-12569.487	-12569.48662	4.095E-12
20	Michalewicz 2	MS	2	[0, π]	-1.8013	-1.801303	-1.801303	9.033E-16
21	Michalewicz 5	MS	5	[0, π]	-4.687658	-4.687658	-4.687658	2.628E-15
22	Michalewicz 10	MS	10	[0, π]	-9.66015	-9.660151	-9.659496	0.001698
23	Schaffer	MN	2	[-100, 100]	0	0	0.003886	0.004841
24	Six Hump Camel Back	MN	2	[-5, 5]	-1.0316	-1.031628	-1.031628	4.516E-16
25	Bohachevsky 2	MN	2	[-100, 100]	0	0	0	0
26	Bohachevsky 3	MN	2	[-100, 100]	0	0	0	0
27	Shubert	MN	2	[-10, 10]	-186.7309	-186.73091	-186.730908	3.095E-13
28	Goldstein-Price	MN	2	[-2, 2]	3	3	3	1.860E-15
29	Kowalik	MN	4	[-5, 5]	0.00031	0.000307	0.000651	0.000259
30	Shekel 5	MN	4	[0, 10]	-10.1532	-10.1532	-10.153199	6.621E-15
31	Shekel 7	MN	4	[0, 10]	-10.4029	-10.402941	-10.227136	0.962917
32	Shekel 10	MN	4	[0, 10]	-10.5364	-10.53641	-10.356145	0.987348
33	Perm	MN	4	[-4, 4]	0	0.000120	0.012280	0.017328
34	Power Sum	MN	4	[0, 4]	0	6.911E-06	0.000565	0.001077
35	Hartman 3	MN	3	[0, 1]	-3.86278	-3.862782	-3.862782	2.710E-15
36	Hartman 6	MN	6	[0, 1]	-3.32237	-3.322368	-3.322368	6.545E-16
37	Griewank	MN	30	[-600, 600]	0	0	0.027246	0.027248
38	Ackley	MN	30	[-32, 32]	0	3.181E-10	6.597E-10	2.562E-10
39	Langerman5	MN	5	[0,10]	-1.5	-1.499999	-1.244818	0.320539
40	FletcherPowell2	MN	2	$[-\pi, \pi]$	0	0	6.731E-28	1.979E-27
41	FletcherPowell5	MN	5	$[-\pi, \pi]$	0	3.155E-26	0.005558	0.015088
42	FletcherPowell10	MN	10	$[-\pi, \pi]$	0	0.003689	15.085456	19.737312

Se observa en los resultados de la Tabla 2, que el valor mínimo promedio obtenido por el algoritmo, en las funciones unimodales y separables evaluadas, se acerca bastante a la solución teórica, siendo la precisión del orden de 1×10^{-18} en la función Sphere y del 1×10^{-19} en la función Sum Squares; además la desviación estándar, es aproximadamente de 1×10^{-18} para estas funciones. Por lo tanto se evidencia que la metaheurística, tiene un excelente desempeño en funciones unimodales separables.

Observando la parte superior de la Figura 3.1, donde se encuentran las curvas de convergencia de estas dos funciones; se ve que el algoritmo converge rápidamente y se concentra en la explotación de las mejores respuestas; lo cual es deseado en funciones unimodales. Según los resultados numéricos hallados y la capacidad de convergencia del algoritmo en este tipo de problemas; se puede concluir que AOOA se desempeñará de manera eficiente en problemas de optimización que sean unimodales y separables.

En funciones unimodales no separables, el algoritmo obtiene excelentes resultados de valor mínimo promedio y desviación estándar, para el 63.6% de los problemas considerados de este tipo; dentro de este porcentaje se encuentran las funciones Beale, Easom, Matyas, Zakharov, Trid 6, Trid 10 y Schwefel 2.22, ver Tabla 2. Sin embargo en funciones como Powell, Rosenbrock, Colville y Dixon Price, que también son unimodales y no separables, únicamente los valores mínimos obtenidos son aceptables; lo que se ratifica al analizar los valores del mínimo promedio y desviación estándar logrados en estas funciones, como se expone en la Tabla 2.

Estos resultados no implican necesariamente, que la técnica se va a comportar de manera inadecuada en funciones unimodales no separables; en cambio sí se puede asegurar que el algoritmo presenta baja repetitividad de buenos resultados, en funciones con valle de curvatura estrecho; como lo son las funciones Rosenbrock, Colville y Dixon Price; en las cuales el óptimo global de la función, fue logrado solo en algunas ocasiones, ver Tabla 2.

En relación a las funciones multimodales separables, el algoritmo de optimización del pulpo artificial, alcanza el valor mínimo teórico de la función, para el 88.88% de las funciones consideradas en este estudio, con una desviación estándar en promedio del 8.88×10^{-11} ; lo cual es un excelente resultado. Para el 11.11% de las funciones evaluadas, porcentaje conformado por el problema de prueba Michalewicz 10, su comportamiento es adecuado; ya que el mínimo promedio obtenido para esta función, posee con el mínimo teórico una diferencia de 6.57×10^{-4} , valor que consigue con una desviación estándar de 1.698×10^{-3} .

En la Figura 3.2, se presentan las curvas de convergencia obtenidas para las funciones Bohachevsky 1, Rastrigin, Schwefel y Michalewicz 10. En las tres últimas funciones, se aprecia como el algoritmo explora eficientemente el espacio de búsqueda y converge prácticamente al final de las iteraciones; demostrando que el algoritmo posee una magnífica habilidad para explorar el espacio de búsqueda, evitando quedar atrapado en óptimos locales.

Al considerar la capacidad del algoritmo para explorar eficazmente el espacio de búsqueda y los buenos resultados obtenidos en esta clase de funciones, se concluye que AOOA, se desempeñará de manera adecuada en la mayoría de problemas de optimización con características multimodales y separables.

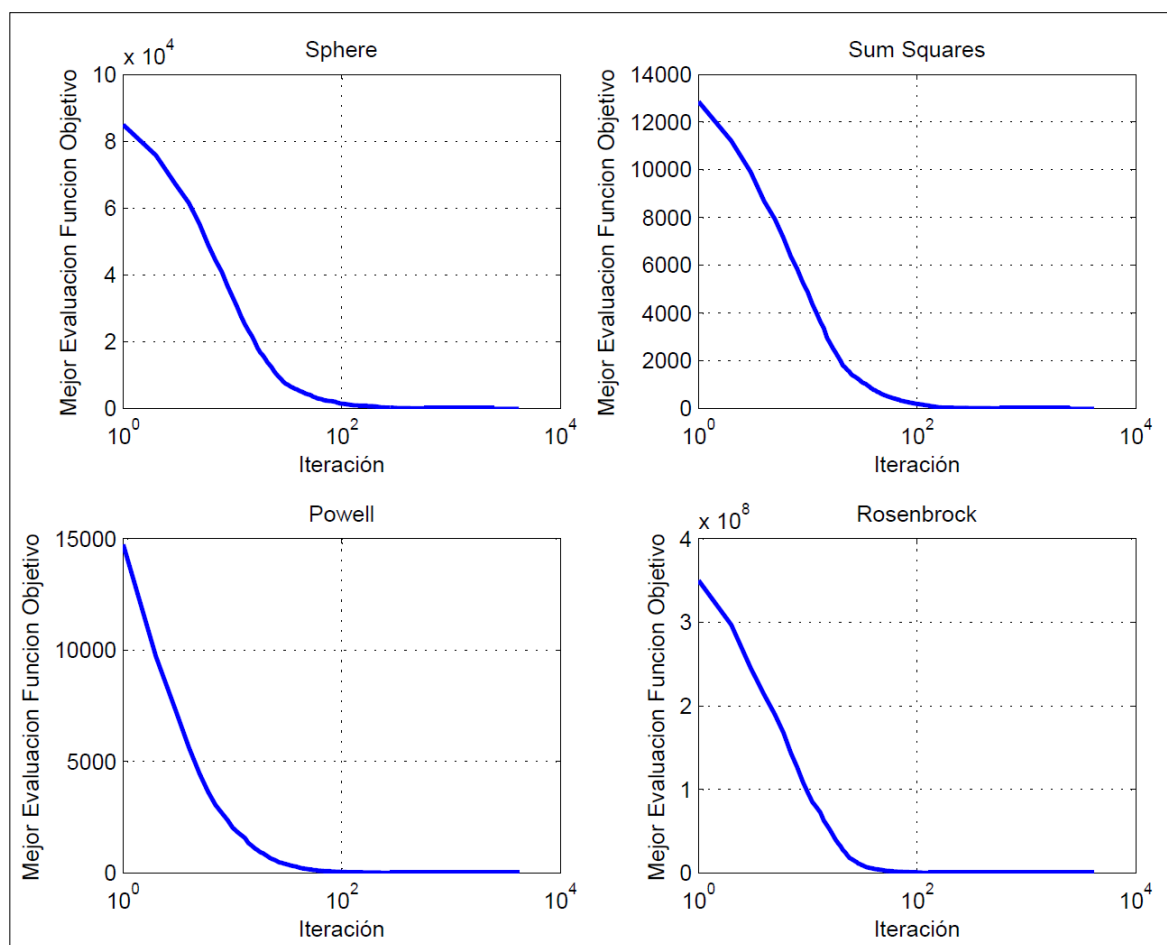


Figura 3.1 Curvas de convergencia de las funciones Sphere, Sum Squares, Powell y Rosenbrock. Las curvas muestran el mejor valor promedio obtenido durante cada iteración en las 30 ejecuciones del algoritmo.

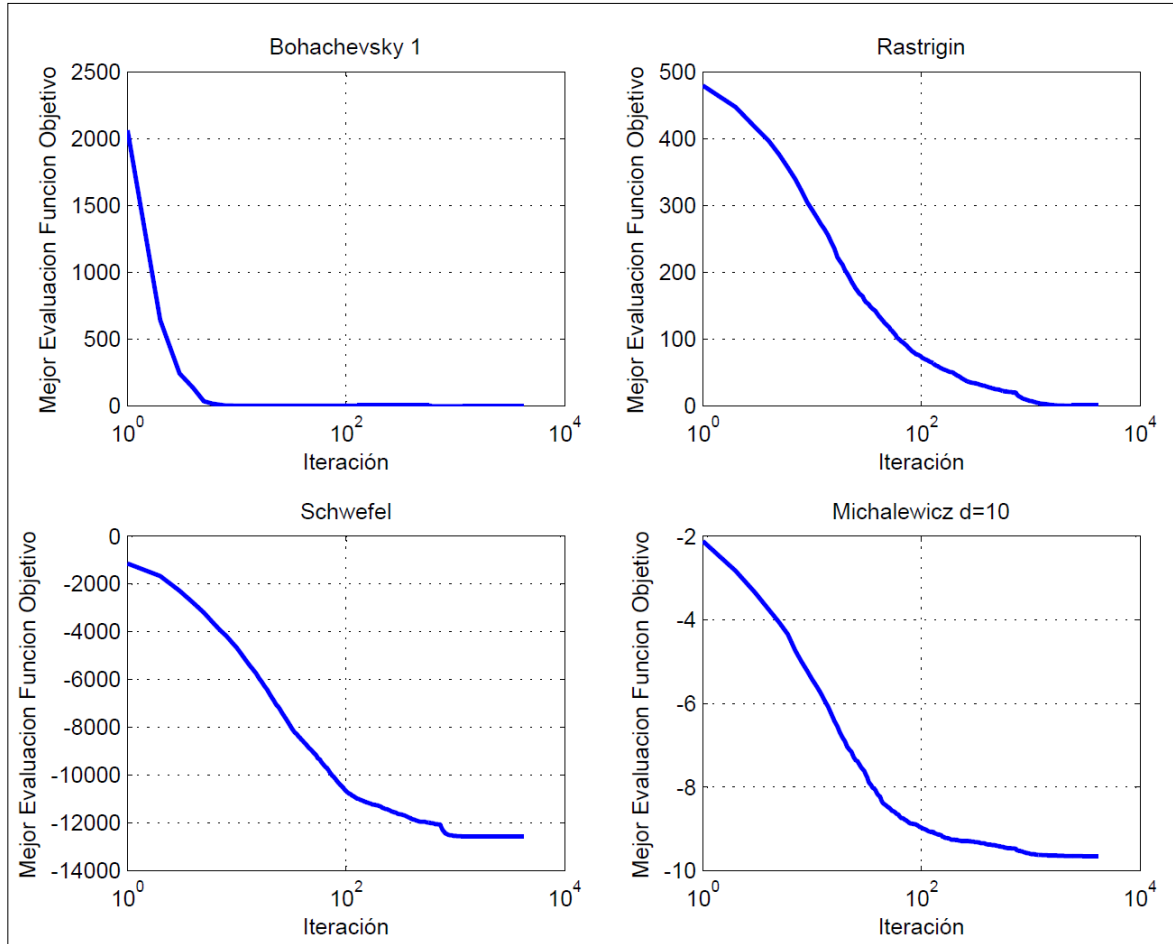


Figura 3.2 Curvas de convergencia de las funciones Bohachevsky 1, Rastrigin, Schwefel y Michalewicz 10. Las curvas muestran el mejor valor promedio obtenido durante cada iteración en las 30 ejecuciones del algoritmo.

La técnica metaheurística se desempeña correctamente en las funciones multimodales y no separables, Six Hump Camel Back, Bohachevsky 2, Bohachevsky 3, Shubert, GoldStein-Price, Shekel 5, Hartman 3, Hartman 6, Ackley y Fletcher Powell 2. Donde el algoritmo alcanza el valor mínimo de estos problemas, con una desviación estándar en promedio de 2.57×10^{-11} . En las funciones Kowalik y Power sum, se logran resultados adecuados, con una precisión en el valor mínimo promedio de 6.51×10^{-4} y 5.65×10^{-4} con desviaciones estándar de 2.59×10^{-4} y 1.08×10^{-3} respectivamente, ver Tabla 2.

Acerca de las funciones Schaffer, Shekel 7, Shekel 10, Perm, Griewank, Langerman 5 y Fletcher Powell 5; se puede mencionar que los valores mínimos obtenidos llegan al óptimo y en el caso de Fletcher Powell 10 bastante cerca. No obstante las desviaciones estándar y

los valores mínimos promedio conseguidos, revelan la falta de repetitividad en la obtención de buenos resultados por el algoritmo AOOA, para estas funciones.

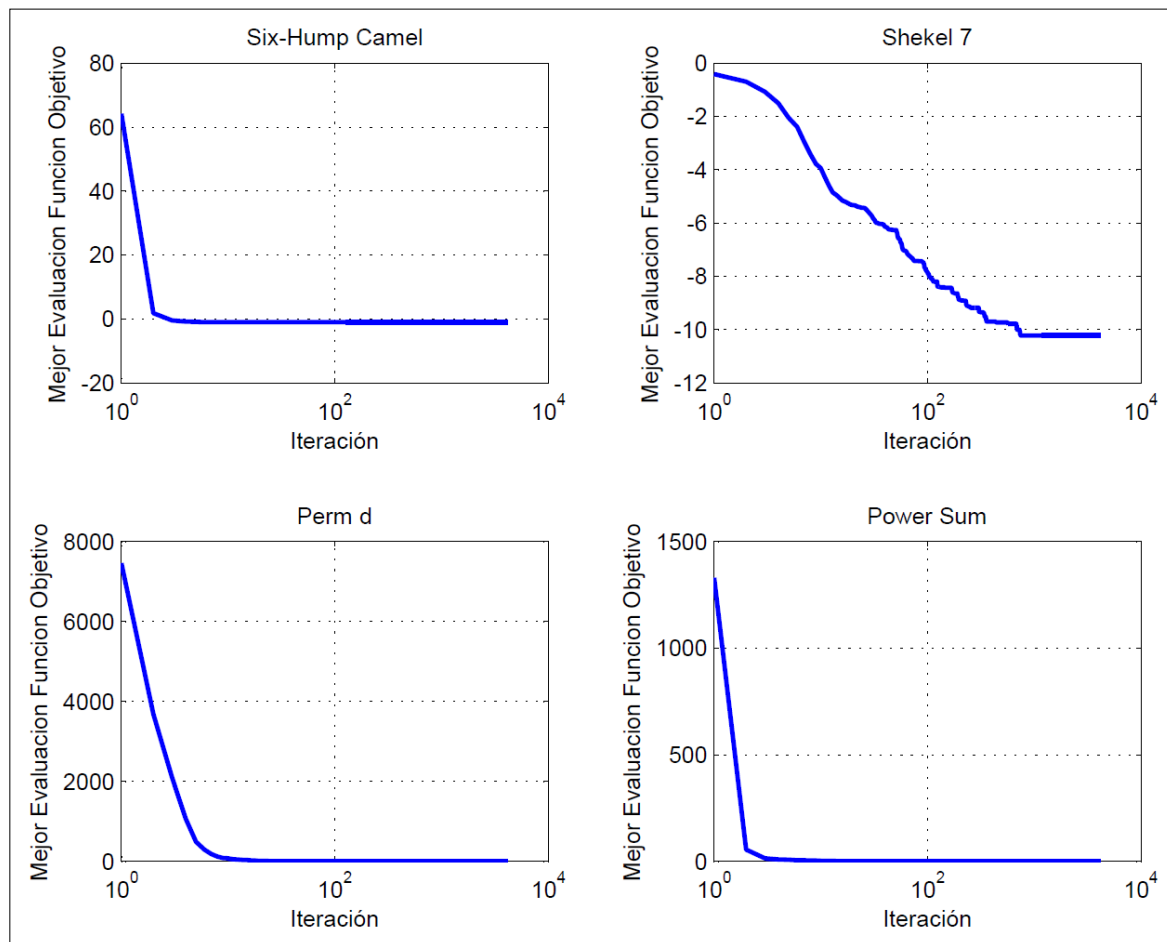


Figura 3.3 Curvas de convergencia de las funciones Six-Hump Camel, Shekel 7, Perm, y Power sum. Las curvas muestran el mejor valor promedio obtenido durante cada iteración en las 30 ejecuciones del algoritmo.

La conducta de exploración eficiente del espacio de búsqueda, que presenta el algoritmo en funciones multimodales separables, de alta o media dimensión; sucede también en las funciones multimodales no separables, Shekel 7, Langerman 5 y Ackley; donde el algoritmo analiza una gran variedad de respuestas antes de converger, como se observa en la Figura 3.3 y Figura 3.4. Las funciones Six-Hump Camel, Perm y Power sum, a pesar de ser multimodales, son funciones de baja dimensión, por lo tanto presentan una convergencia más rápida.

Al realizar un balance del desempeño general del algoritmo, cabe resaltar que con un límite de cien mil evaluaciones de la función objetivo, AOOA entrega resultados de calidad en

valores mínimo promedio obtenidos, en el 71.42% de las funciones de prueba consideradas; asegurando la repetitividad de estas soluciones, por los valores de desviación estándar obtenidos. Además el valor mínimo obtenido con la técnica, llega al óptimo con precisión mínima de 1×10^{-3} en el 92.85% de las funciones de prueba consideradas. Permitiendo concluir que el algoritmo posee un buen desempeño en gran variedad de funciones; el cual se puede mejorar aún más, al realizar suficientes repeticiones de la ejecución del algoritmo.

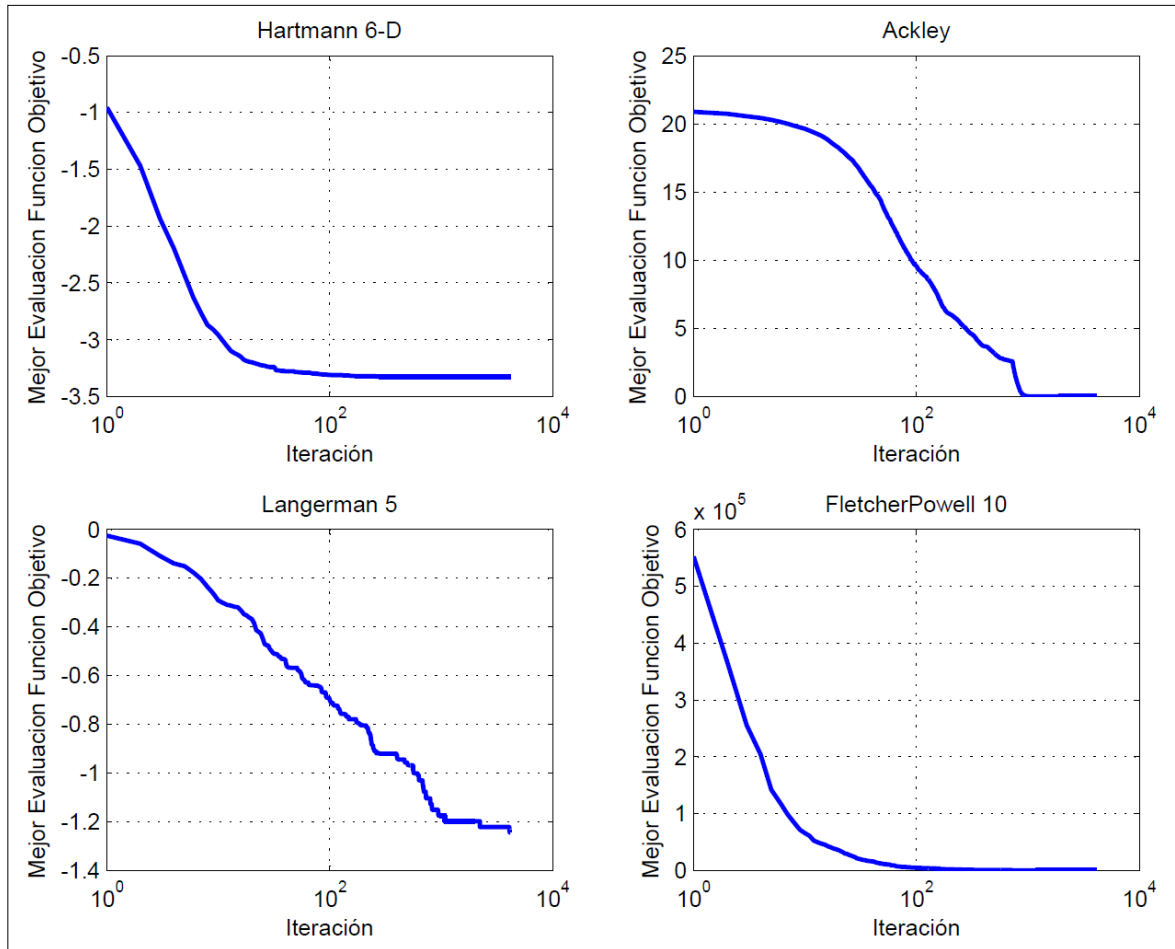


Figura 3.4 Curvas de convergencia de las funciones Hartman 6-D, Ackley, Langerman 5 y Fletche Powell 10. Las curvas muestran el mejor valor promedio obtenido durante cada iteración en las 30 ejecuciones del algoritmo.

Gracias a los resultados obtenidos en funciones con características especiales, a continuación se realiza una descripción de algunas propiedades de AOOA.

En primer lugar se tiene la habilidad que le permite encontrar el óptimo en funciones planas, las cuales no brindan información de la dirección de búsqueda, como lo son Matyas y

PowerSum [90], [92]. Otra característica, es el mecanismo de exploración local que posee; el cual permite acercarse a la solución en funciones con el óptimo global cerca de óptimos locales, como se corrobora al observar los resultados de las funciones Kowalik, Perm y Schaffer [92], [91]. Posee además la capacidad de hallar el óptimo global en zonas angostas y pequeñas en comparación al espacio de búsqueda, como en las funciones Easom y Michalewicz 10 [90], [91], [92].

El algoritmo AOOA también supera los problemas de escalabilidad, donde existen varios órdenes de magnitud entre las variables y la superficie de la función, como en GoldStein-Price y Trid [90], [92]. Por último se puede mencionar la falta de destreza de AOOA para hallar el óptimo en funciones con un valle de curvatura estrecho, como las funciones Rosenbrock, Colville y Dixon Price [90], [91]; en las cuales el algoritmo posee baja repetitividad en la obtención de buenos resultados.

3.3 Evaluación de desempeño con respecto a otros algoritmos

A continuación se realizan dos pruebas comparativas, donde se coteja AOOA con otras técnicas de optimización metaheurística; esto con el fin de evaluar el desempeño del algoritmo propuesto en la resolución de problemas de optimización, frente a algoritmos clásicos de optimización metaheurística.

3.3.1 Prueba comparativa 1 AOOA vs GA, DE, PSO y ABC

Para la realización de esta prueba comparativa, se utilizaron los resultados obtenidos en un estudio comparativo publicado por Karaboga y Akay [91]. Estas pruebas fueron realizadas con un tamaño de población de cincuenta (50) individuos, para un máximo de quinientas mil (500.000) evaluaciones de la función objetivo; las pruebas fueron realizadas 30 veces y luego fueron promediadas. Los resultados por debajo de 1×10^{-12} son considerados como cero; esto con el fin de facilitar las comparaciones.

Las configuraciones en los parámetros de los algoritmos GA, DE, PSO y ABC, se encuentran en el estudio realizado por Karaboga [91]. Para el algoritmo de optimización del pulpo artificial, se utilizará un valor Pr igual a 0.175, como se justificará en la selección de

parámetros adecuados para la metaheurística, página 78; la cantidad de ventosas por brazo es ns igual a 6, con el fin de obtener una población por iteración cercana a cincuenta (50).

Tabla 3. Resultados estadísticos de las 30 evaluaciones de los algoritmos GA [91], PSO [91], DE [91], ABC [91] y AOOA. Prom.: promedio de los mejores valores obtenidos en cada evaluación, D. Estd: desviación estándar. En **negrita** valores mínimos.

Nombre	Mínimo		GA	PSO	DE	ABC	AOOA
Sphere	0	Prom.	1.11E+03	0	0	0	0
		D. Estd.	74.214474	0	0	0	0
Sum Squares	0	Prom.	1.48E+02	0	0	0	0
		D. Estd.	12.4092893	0	0	0	0
Beale	0	Prom.	0	0	0	0	0
		D. Estd.	0	0	0	0	0
Easom	-1	Prom.	-1	-1	-1	-1	-1
		D. Estd.	0	0	0	0	0
Matyas	0	Prom.	0	0	0	0	0
		D. Estd.	0	0	0	0	0
Colville	0	Prom.	0.014938	0	0.040912	0.092967	7.715E-08
		D. Estd.	0.007364	0	0.081979	0.066277	1.018E-07
Trid 6	-50	Prom.	-49.9999	-50	-50	-50	-50
		D. Estd.	2.25E-05	0	0	0	0
Trid 10	-210	Prom.	-209.476	-210	-210	-210	-210
		D. Estd.	0.193417	0	0	0	0
Zakharov	0	Prom.	0.013355	0	0	0.000247	0
		D. Estd.	0.004532	0	0	0.00183	0
Powell	0	Prom.	9.703771	0.00011	2.17E-07	0.003134	0.000246
		D. Estd.	1.547983	0.00016	1.36E-07	0.000503	3.925E-05
Schwefel 2.22	0	Prom.	11.0214	0	0	0	0
		D. Estd.	1.386856	0	0	0	0
Rosenbrock	0	Prom.	1.96E+05	15.088617	18.203938	0.088770	16.0395
		D. Estd.	3.85E+04	24.170196	5.036187	0.07739	25.146755
Dixon-Price	0	Prom.	1.22E+03	0.666666	0.666666	0	0.622221
		D. Estd.	2.66E+02	1.00E-08	1.00E-09	0	0.169138
Foxholes	0.998003	Prom.	0.998004	0.998003	0.998003	0.998003	0.998003
		D. Estd.	0	0	0	0	0
Branin	0.397887	Prom.	0.397887	0.397887	0.397887	0.397887	0.397887
		D. Estd.	0	0	0	0	0
Bohachevsky 1	0	Prom.	0	0	0	0	0
		D. Estd.	0	0	0	0	0
Booth	0	Prom.	0	0	0	0	0
		D. Estd.	0	0	0	0	0
Rastrigin	0	Prom.	52.92259	43.977136	11.716728	0	0
		D. Estd.	4.56486	11.728676	2.538172	0	0
Schwefel	-12569.4866	Prom.	-11593.4	-6909.1359	521.849292	-12569.487	-12569.4866
		D. Estd.	93.25424	457.957783	95.276209	0	0
Michalewicz 2	-1.8013	Prom.	-1.8013	-1.572869	-1.801303	-1.801303	-1.801303
		D. Estd.	0	0.11986	0	0	0
Michalewicz 5	-4.687658	Prom.	-4.64483	-2.490872	-4.683482	-4.687658	-4.687658
		D. Estd.	0.09785	0.256952	0.012529	0	0
Michalewicz 10	-9.66015	Prom.	-9.49683	-4.00718	-9.591151	-9.660151	-9.660151
		D. Estd.	0.141116	0.502628	0.064205	0	0
Schaffer	0	Prom.	0.004239	0	0	0	0
		D. Estd.	0.004763	0	0	0	0
Six Hump Camel Back	-1.0316	Prom.	-1.03163	-1.031628	-1.031628	-1.031628	-1.031628
		D. Estd.	0	0	0	0	0

Tabla 3. Continuación. Resultados estadísticos de las 30 evaluaciones de los algoritmos GA [91], PSO [91], DE [91], ABC [91] y AOOA. Prom.: promedio de los mejores valores obtenidos en cada evaluación, D. Estd: desviación estándar. En negrita valores mínimos.

Bohachevsky 2	0	Prom.	0.06829	0	0	0	0
		D. Estd.	0.078216	0	0	0	0
Bohachevsky 3	0	Prom.	0	0	0	0	0
		D. Estd.	0	0	0	0	0
Shubert	-186.7309	Prom.	-186.731	-186.7309	-186.7309	-186.7309	-186.7309
		D. Estd.	0	0	0	0	0
GoldStein-Price	3	Prom.	5.520611	3	3	3	3
		D. Estd.	5.870093	0	0	0	0
Kowalik	0.00031	Prom.	0.005615	0.00049	0.000426	0.000426	0.000316
		D. Estd.	0.008171	0.000366	0.000273	6.04E-05	4.781E-05
Shekel 5	-10.1532	Prom.	-5.66052	-2.087007	-10.1532	-10.1532	-10.1532
		D. Estd.	3.866737	1.17846	0	0	0
Shekel 7	-10.4029	Prom.	-5.34409	-1.989871	-10.40294	-10.40294	-10.40294
		D. Estd.	3.517134	1.420602	0	0	0
Shekel 10	-10.5364	Prom.	-3.82984	-1.879675	-10.53641	-10.53641	-10.5364
		D. Estd.	2.451956	0.432476	0	0	0
Perm	0	Prom.	0.302671	0.036051	0.024006	0.041105	0.006563
		D. Estd.	0.193254	0.048927	0.046032	0.023056	0.006301
Power Sum	0	Prom.	0.010405	11.390447	0.000142	0.002946	0.000122
		D. Estd.	0.009077	7.3558	0.000145	0.002289	0.00015
Hartman 3	-3.86278	Prom.	-3.86278	-3.633352	-3.862782	-3.862782	-3.862782
		D. Estd.	0	0.116937	0	0	0
Hartman 6	-3.32237	Prom.	-3.29822	-1.859129	-3.226881	-3.321995	-3.322368
		D. Estd.	0.05013	0.439958	0.047557	0	0
Griewank	0	Prom.	10.63346	0.017391	0.001479	0	0.014344
		D. Estd.	1.161455	0.020808	0.002958	0	0.015668
Ackley	0	Prom.	14.67178	0.164622	0	0	0
		D. Estd.	0.178141	0.493867	0	0	0
Langerman5	-1.5	Prom.	-0.96842	0.504857	-1.499999	-0.93815	-1.391099
		D. Estd.	0.287548	0.213626	0	0.000208	0.221734
Fletcher Powell 2	0	Prom.	0	0	0	0	0
		D. Estd.	0	0	0	0	0
Fletcher Powell 5	0	Prom.	0.004303	1457.88344	5.988783	0.173549	4.092E-07
		D. Estd.	0.009469	1269.362389	7.334731	0.068175	1.295E-06
Fletcher Powell 10	0	Prom.	29.57348	1364.45555	781.55028	8.092742	1.660579
		D. Estd.	16.021078	1325.379655	1048.813487	1.477526	4.228131
<i>Total</i>			13	20	27	31	35

De los resultados obtenidos en la Tabla 3, se puede mencionar, que en cuanto a las funciones unimodales y separables, todos los algoritmos muestran un buen desempeño, a excepción de GA. Debido tal vez a la codificación binaria empleada para la obtención de resultados en el estudio de Karaboga [91]; ya que una configuración de punto flotante sería más consistente y podría mejorar la precisión en problemas específicos [94].

En las funciones unimodales no separables, todos los algoritmos considerados poseen buenos resultados en las funciones, Beale, Easom y Matyas; en las funciones Trid 6, Trid 10

y Schwefel, todos menos GA alcanzan el óptimo de la función. Se puede destacar además que en la función Zakharov, fallan únicamente GA y ABC; los problemas Rosenbrock y Dixon-Price son resueltos de manera eficaz únicamente por ABC. La función Powell sólo es resuelta de manera adecuada por DE y la función Colville únicamente por PSO.

En cuanto a las funciones multimodales y separables, todos los algoritmos considerados entregan buenos resultados en las funciones, Foxholes, Branin, Bohachevsky 1 y Booth; en la función Michalewicz 2 todos a excepción de PSO logran el óptimo global. Sobresale además al observar los resultados de la Tabla 3 que en las funciones Rastrigin, Schwefel, Michalewicz 5 y Michalewicz 10, únicamente ABC Y AOOA entregan resultados adecuados.

Por último al observar los resultados para funciones multimodales no separables, se observa que todos los algoritmos alcanzan el valor mínimo de las funciones, Schaffer, Six-Hump Camel Back, Bohachevsky 3 y Shubert; todos los algoritmos a excepción de GA, ostentan un buen desempeño en las funciones Bohachevsky 2 y Goldstein-Price. En la función Hartman 3, PSO es el único de los algoritmos que no consigue el resultado esperado.

Se resalta de los datos en la Tabla 3, que para las funciones Shekel 5, Shekel 7, Shekel 10 y Ackley, solamente los algoritmos ABC, DE y AOOA encuentran el óptimo de la función. La función Griewank es solucionada aceptablemente sólo por ABC y la función Langerman 5 solo por DE. En los problemas Kowalik, Hartman 6, Fletcher-Powell 5, Perm y Power Sum, exclusivamente AOOA alcanza el valor óptimo. En el problema Fletcher-Powell 10, aunque ninguno de los algoritmos llega al valor mínimo de la función con una precisión apropiada; AOOA es el que tiene los mejores resultados.

De los anteriores resultados se concluye, que AOOA entrega buenos resultados en las funciones unimodales tanto separables como no separables; sin embargo sus resultados en algunas de estas funciones son superados por ABC, PSO y DE. En las funciones multimodales separables se observa como AOOA destaca junto a ABC, donde los otros algoritmos no encuentran resultados adecuados. Sin embargo el verdadero potencial de AOOA se observa en las funciones multimodales no separables; donde AOOA se desempeña mejor que los otros algoritmos aquí considerados.

En el conteo general de la Tabla 3, se ve que AOOA obtiene el óptimo para 35 de las 42 funciones consideradas, seguido con 31 por ABC, 27 por DE, 20 por PSO y 13 por GA;

confirmando que AOOA es una técnica competitiva respecto a otras técnicas de optimización metaheurística.

3.3.2 Prueba comparativa 2 AOOA vs estrategias evolutivas y ABC

En este numeral se realiza una prueba comparativa entre AOOA, ABC y algunas estrategias evolutivas (CES, ESLAT y CMA-ES). Los datos para ABC, son tomados del estudio comparativo realizado por Karaboga y Akay [91]; los resultados de las estrategias evolutivas, se tomaron del artículo publicado por Hedar y Fukushima [95].

Estas pruebas fueron realizadas para ABC, con un tamaño de población de veinte (20) individuos y un límite de abandono $SN \cdot D$; donde SN es el número de fuentes disponibles y D es la dimensión del problema. Para AOOA el valor Pr es igual a 0.175 y ns es 3. Las configuraciones generales de las estrategias evolutivas, se encuentran en el trabajo de A. Hedar y M. Fukushima [95].

Para la realización de esta prueba comparativa, el número límite de evaluaciones de la función objetivo es de cien mil (100,000); sin embargo si se lograba una precisión de 1×10^{-3} en la respuesta, la ejecución del algoritmo era finalizada antes de llegar al límite de evaluaciones permitido. Cada prueba se realizó 50 veces y las mejores respuestas de cada función se promediaron; como se especifica en [91] y [95]. Los resultados de este estudio se encuentran consignados en la Tabla 4, Tabla 5 y Tabla 6.

En los resultados de la Tabla 4, se pueden observar los valores mínimos obtenidos para cada función. En negrita están los mejores valores de cada algoritmo en comparación a las otras técnicas; donde AOOA y ABC están en primer lugar con 9 soluciones, las estrategias evolutivas poseen solo 5 funciones con calidad sobresaliente.

Realizando una comparación más detallada de los datos en la Tabla 4, se observa como AOOA es mejor en 7 problemas (Foxholes, Branin, Rastrigin, Schwefel, Shekel 5, Hartman 3 y Hartman 6) que CES, a su vez CES solo es mejor que AOOA en 3 funciones (Sphere, Schwefel 2.22 y Ackley). En comparación con ESLAT, se observa como AOOA la supera en 5 funciones (Foxholes, Rastrigin, Schwefel, Shekel 5 y Hartman 6) mientras que ESLAT es mejor en únicamente en 1 problema (Kowalik). Al cotejar los resultados de AOOA y CMA-ES, se observa que el algoritmo de optimización del pulpo artificial, es mejor en 6 funciones

(Foxholes, Rastrigin, Schwefel, GoldStein-Price Shekel 5 y Hartman 6) que CMA-ES; mientras este solo lo supera en 2 (Griewank y Rosenbrock). Por último se observa que ABC y AOOA son similares en cuanto a calidad de resultados; ya que AOOA supera a ABC en las funciones Schwefel y Shekel 5 y a su vez ABC lo supera en Shekel 7 y Shekel 10.

Tabla 4. Resultados estadísticos de las 50 evaluaciones de los algoritmos CES [95], ESLAT [95], CMA-ES [95], ABC [91] y AOOA. Prom.: promedio de los mejores valores obtenidos, D. Estd: desviación estándar. En negrita valores mínimos.

Nombre	Mínimo		CES	ESLAT	CMA-ES	ABC	AOOA
Sphere	0	Prom.	1.70E-26	2.00E-17	9.70E-23	7.57E-04	9.00E-04
		D. Estd.	1.10E-25	2.90E-17	3.80E-23	2.48E-04	1.42E-04
Schwefel 2.22	0	Prom.	8.10E-20	3.80E-05	4.20E-11	8.95E-04	9.23E-04
		D. Estd.	3.60E-19	1.60E-05	7.10E-23	1.27E-04	9.07E-05
Rosenbrock	0	Prom.	27.65	1.93	0.4	9.36E-01	6.90E+01
		D. Estd.	0.51	3.35	1.2	1.76E+00	1.02E+02
Foxholes	0.998003	Prom.	2.2	1.77	10.44	9.98E-01	9.98E-01
		D. Estd.	2.43	1.37	6.87	3.21E-04	2.68E-04
Branin	0.397887	Prom.	0.401	0.398	0.398	0.3985	0.39847756
		D. Estd.	3.60E-03	1.00E-13	1.40E-15	3.27E-04	2.68E-04
Rastrigin	0	Prom.	13.38	4.65	51.78	4.66E-04	8.46E-04
		D. Estd.	43.15	5.67	13.56	3.44E-04	2.03E-04
Schwefel	-12569.49	Prom.	-8.00E+93	-2.30E+15	-7637.14	-12563.6734	-12569.4857
		D. Estd.	4.90E+94	5.70E+15	895.6	2.36E+01	1.42E-04
Six Hump Camel Back	-1.0316	Prom.	-1.031	-1.0316	-1.0316	-1.031	-1.03112066
		D. Estd.	1.20E-03	9.70E-14	7.70E-16	3.04E-04	2.93E-04
GoldStein- Price	3	Prom.	3.007	3.00E+00	14.34	3	3.0005052
		D. Estd.	1.20E-02	5.80E-14	25.05	3.09E-04	2.96E-04
Kowalik	0.00031	Prom.	1.30E-03	8.10E-04	1.50E-03	1.18E-03	1.20E-03
		D. Estd.	6.30E-04	4.10E-04	4.20E-03	1.45E-04	1.29E-04
Shekel 5	-10.1532	Prom.	-5.72	-8.49	-5.86	-10.151	-10.1525528
		D. Estd.	2.62	2.76	3.6	1.17E-02	2.85E-04
Shekel 7	-10.4029	Prom.	-6.09	-8.79	-6.58	-10.402	-10.1904534
		D. Estd.	2.63	2.64	3.74	3.11E-04	1.05E+00
Shekel 10	-10.5364	Prom.	-6.42	-9.65	-7.03	-10.535	-10.186421
		D. Estd.	2.67	2.06	3.74	2.02E-03	1.41E+00
Hartman 3	-3.86278	Prom.	-3.8613	-3.8628	-3.8628	-3.862	-3.86219674
		D. Estd.	1.20E-03	2.90E-13	4.80E-16	2.77E-04	2.81E-04
Hartman 6	-3.32237	Prom.	-3.24	-3.31	-3.28	-3.322	-3.32162611
		D. Estd.	5.80E-02	3.30E-02	5.80E-02	1.35E-04	2.11E-04
Griewank	0	Prom.	6.00E-14	1.40E-03	7.40E-04	8.37E-04	2.32E-02
		D. Estd.	4.20E-13	4.70E-03	2.70E-03	1.38E-03	2.47E-02
Ackley	0	Prom.	6.00E-13	1.80E-08	6.90E-12	7.81E-04	9.47E-04
		D. Estd.	1.70E-12	5.40E-09	1.30E-12	1.83E-04	7.07E-05
Total			5	5	5	9	9

Tabla 5. Costo de la solución, en función del número promedio de las evaluaciones de la función objetivo, para los algoritmos CES [95], ESLAT [95], CMA-ES [95], ABC [91] y AOOA.

No	Nombre	CES Promedio	ESLAT Promedio	CMA-ES Promedio	ABC Promedio	AOOA Promedio
1	Sphere	69724	69724	10721	9264	25317
11	Schwefel 2.22	60859	60859	12145	12991	30363
12	Rosenbrock	66609	66609	55821	100000	100000
14	Foxholes	1305	1305	540	1046	955
15	Branin	1257	1257	594	530	293
18	Rastrigin	53880	53880	10079	26731	47749
19	Schwefel	61704	61704	6621	64632	29879
24	Six Hump Camel Back	1306	1306	619	342	231
28	Goldstein-Price	1201	1201	2052	15186	657
29	Kowalik	2869	2869	13434	6120	5296
30	Shekel 5	2338	2338	1246	6069	6658
31	Shekel 7	2468	2468	1267	7173	10554
32	Shekel 10	2410	2410	1275	15392	13939
35	Hartman 3	1734	1734	996	4747	404
36	Hartman 6	3816	3816	2293	1583	2715
37	Griewank	71044	71044	10522	36151	84542
38	Ackley	58909	58909	10654	16616	31895
Total		0	1	10	2	4

Tabla 6. Tasa de éxito (%) de los algoritmos ESLAT [95], CMA-ES [95], ABC [91] y AOOA.

No	Nombre	ESLAT Promedio	CMA-ES Promedio	ABC Promedio	AOOA Promedio
1	Sphere	100	100	100	100
11	Schwefel 2.22	100	100	100	100
12	Rosenbrock	70	90	0	0
14	Foxholes	60	0	100	100
15	Branin	100	100	100	100
18	Rastrigin	40	0	100	100
19	Schwefel	0	0	86	100
24	Six Hump Camel Back	100	100	100	100
28	Goldstein-Price	100	78	100	100
29	Kowalik	94	88	100	100
30	Shekel 5	72	40	98	100
31	Shekel 7	72	48	100	93.33
32	Shekel 10	84	52	96	93.33
35	Hartman 3	100	100	100	100
36	Hartman 6	94	48	100	100
37	Griewank	90	92	96	16.66
38	Ackley	100	100	100	100
Total		7	7	14	13

En cuanto a los datos de la Tabla 5, la cual muestra el desempeño de los algoritmos en función de la cantidad de evaluaciones realizadas a la función objetivo, para obtener resultados con precisión de 1×10^{-3} . Se observa como CMA-ES posee el menor costo computacional; al requerir menos evaluaciones de la función objetivo que los demás algoritmos en 10 de las funciones consideradas, seguido por AOOA con 4 funciones, ABC con 2 y ESLAT con 1.

Sin embargo se observa en la Tabla 6 que la tasa de éxito, la cual mide el porcentaje de ejecuciones en las cuales hay convergencia, que la estrategia CMA-ES solo es buena generalmente para funciones unimodales. Adicionalmente se ve en la Tabla 6, que los algoritmos con mejor convergencia son ABC y AOOA con 14 y 13 resultados superiores a los demás algoritmos respectivamente; donde AOOA posee 100% de convergencia en 13 funciones mientras que ABC solo en 12; de donde se puede ver el excelente desempeño de AOOA con respecto a las otras técnicas consideradas.

Con respecto a la tasa de éxito de AOOA, para las 17 funciones estudiadas, se puede observar como para 15 de ellas tiene alta tasa de convergencia; sin embargo se corrobora nuevamente la falta de destreza del algoritmo, para resolver problemas con valle de curvatura estrecho; ya que la tasa de éxito para la función de Rosenbrock es de cero (0). Se ve además como para la función Griewank, la tasa de éxito es bastante baja; lo que explica los resultados previamente obtenidos en la Tabla 2 y Tabla 3.

Al reunir los resultados encontrados en ambas pruebas comparativas, se ve que el algoritmo de optimización del pulpo artificial, es adecuado para el desarrollo de problemas de optimización; cubriendo eficientemente gran variedad de tipos de funciones. Adicionalmente los resultados que AOOA encuentra, está al nivel de otras técnicas de optimización metaheurística conocidas.

3.4 Selección de parámetros adecuados en AOOA y análisis de sensibilidad de respuesta

Los algoritmos metaheurísticos para su funcionamiento, emplean por lo general parámetros numéricos y en algunos casos categóricos; los cuales influyen en la calidad de las soluciones obtenidas [96]. La determinación adecuada de estos valores, es un aspecto fundamental para obtener una buena configuración del algoritmo; por ello se lleva a cabo un proceso de ajuste de parámetros, el cual usualmente mejora su desempeño [96].

En la mayoría de las técnicas metaheurísticas encontradas en la literatura, este procedimiento se realiza por medio de un método de ensayo y error [96], [97]. Sin embargo existen otros métodos; como por ejemplo el empleado por Hooker [98] o Xu y Kelly [99], quienes utilizan un acercamiento a partir de diseño de experimentos, Coy [97] y Adenso-Díaz [100] utilizan una metodología de superficie de respuesta, Hutter [101] emplea un proceso de búsqueda local; la configuración de parámetros en línea es otra de las opciones para realizar el ajuste, este sistema que funciona con una especie de aprendizaje de máquina, varía algunos parámetros mientras va buscando la respuesta [96].

Debido a que la técnica metaheurística AOOA, posee únicamente dos parámetros a sintonizar, siendo uno de ellos el número de ventosas por brazo ns , el cual influencia directamente la población de la metaheurística y siendo el otro Pr , el coeficiente que controla la explotación y exploración del algoritmo. Se opta por realizar la sintonización de la técnica, por medio de un estudio extensivo de parámetros el cual se expone a continuación.

Inicialmente del grupo total de funciones a evaluar, se toma el subconjunto propuesto en la Tabla 7; las funciones que lo integran poseen la mayor cantidad de diferencias estructurales entre sí; es decir, son una muestra representativa del conjunto inicial descrito en la Tabla 1. Para este caso, se seleccionaron funciones como SumSquares; no diferenciables, como Schwefel 2.22; con forma de valle, tal como Rosenbrock; con múltiples óptimos locales, tales como Rastrigin y Ackley; con crestas empinadas como Michalewicz 10 o en forma de placa como Power sum. Además se incluyeron en este subgrupo, problemas de diferentes tipos según modalidad y separabilidad.

Tabla 7. Conjunto de funciones utilizadas para la sintonización de parámetros

No	Nombre	T	D	Dominio
2	SumSquares	US	30	$[-10, 10]$
11	Schwefel 2.22	UN	30	$[-10, 10]$
12	Rosenbrock	UN	30	$[-30, 30]$
18	Rastrigin	MS	30	$[-5.12, 5.12]$
22	Michalewicz 10	MS	10	$[0, \pi]$
31	Shekel 7	MN	4	$[0, 10]$
34	Power Sum	MN	4	$[0, 4]$
38	Ackley	MN	30	$[-32, 32]$

Para crear las distintas configuraciones de parámetros a examinar, se escogieron los límites de Pr y ns , en los valores donde la técnica se desempeñaba adecuadamente y existía un tiempo adecuado para la ejecución del algoritmo. El intervalo analizado, también está delimitado por los valores que permite la técnica metaheurística; estos son indicados en las ecuaciones (3.1) y (3.2). La variación de cada parámetro se escogió de tal manera que la técnica presentara un cambio considerable en la respuesta. Estos criterios para llevar a cabo el estudio, son similares a los empleados por Coy [97] para la realización de un estudio piloto de parámetros.

$$ns \in \mathbb{z}^+ \quad (3.1)$$

$$0 < Pr < 0.1 \quad (3.2)$$

Para evaluar el algoritmo se utilizaron cien mil (100,000) evaluaciones de la función objetivo y se repitió cada configuración de Pr y ns 30 veces; los resultados se encuentran en la Tabla 8; además se encuentran graficados en la Figura 3.5, Figura 3.6, Figura 3.7 y Figura 3.8, con el fin de poder complementar el análisis.

Conjuntamente a la selección de parámetros, se lleva a cabo un análisis de sensibilidad de respuesta de la técnica metaheurística. Estudio llevado a cabo con el propósito de analizar la influencia que poseen Pr y ns en la respuesta obtenida, la desviación estándar y el tiempo empleado en la ejecución del algoritmo.

Tabla 8. Estudio extensivo de parámetros donde cada configuración fue realizada 30 veces y promediados sus resultados.

Configuración			Sum Squares			Schwefel 2.22		
<i>ns</i>	<i>Pr</i>	<i>It_{max}</i>	Respuesta Promedio	Desviación Estándar	Tiempo Promedio [s]	Respuesta Promedio	Desviación Estándar	Tiempo Promedio [s]
1	0.005	12500	3.404E-18	1.862E-17	3.15	2.357E-10	6.503E-10	3.17
3	0.005	4166	5.764E-22	9.918E-22	2.59	4.385E-12	1.269E-11	2.58
5	0.005	2500	8.597E-22	3.871E-21	2.43	3.347E-13	6.020E-13	2.45
1	0.01	12500	3.795E-20	4.969E-20	3.11	1.932E-10	2.588E-10	3.12
3	0.01	4166	4.730E-21	8.237E-21	2.55	3.769E-12	7.230E-12	2.57
5	0.01	2500	3.812E-21	7.714E-21	2.43	2.898E-12	5.034E-12	2.45
1	0.015	12500	1.659E-16	4.425E-16	3.05	6.508E-10	6.170E-10	3.09
3	0.015	4166	1.821E-19	1.640E-19	2.51	1.105E-10	8.241E-11	2.56
5	0.015	2500	2.967E-19	2.790E-19	2.39	8.749E-11	5.933E-11	2.44
1	0.02	12500	3.517E-15	6.286E-15	3.03	3.280E-08	3.449E-08	3.08
3	0.02	4166	2.143E-16	1.153E-15	2.50	6.716E-10	2.193E-10	2.55
5	0.02	2500	1.011E-16	5.377E-16	2.39	8.843E-10	5.017E-10	2.43
1	0.025	12500	2.020E-13	1.869E-13	2.99	2.445E-07	1.324E-07	3.07
3	0.025	4166	2.800E-14	5.690E-14	2.47	3.066E-08	3.477E-08	2.54
5	0.025	2500	4.494E-14	7.044E-14	2.36	2.595E-08	3.516E-08	2.43
Configuración			Rosenbrock			Rastrigin		
<i>ns</i>	<i>Pr</i>	<i>It_{max}</i>	Respuesta Promedio	Desviación Estándar	Tiempo Promedio [s]	Respuesta Promedio	Desviación Estándar	Tiempo Promedio [s]
1	0.005	12500	9.364E+01	1.087E+02	3.23	2.001E-01	4.827E-01	3.46
3	0.005	4166	9.132E+01	1.383E+02	2.61	6.880E-02	2.521E-01	2.85
5	0.005	2500	2.048E+02	4.185E+02	2.50	7.063E-02	2.523E-01	2.66
1	0.01	12500	8.427E+01	1.471E+02	3.21	6.633E-02	2.524E-01	3.44
3	0.01	4166	6.250E+01	7.945E+01	2.62	3.652E-08	2.000E-07	2.82
5	0.01	2500	8.785E+01	1.544E+02	2.50	5.595E-11	3.014E-10	2.70
1	0.015	12500	8.011E+01	1.117E+02	3.18	1.373E-07	7.519E-07	3.41
3	0.015	4166	6.042E+01	5.286E+01	2.61	2.747E-13	5.047E-13	2.77
5	0.015	2500	6.650E+01	1.064E+02	2.50	4.112E-13	1.114E-12	2.68
1	0.02	12500	5.306E+01	5.710E+01	3.16	7.181E-11	3.526E-10	3.40
3	0.02	4166	6.222E+01	3.393E+01	2.61	9.834E-13	1.210E-12	2.77
5	0.02	2500	6.700E+01	5.512E+01	2.64	9.436E-13	8.151E-13	2.66
1	0.025	12500	6.404E+01	6.045E+01	3.28	7.845E-09	4.280E-08	3.38
3	0.025	4166	5.816E+01	5.804E+01	2.76	5.194E-12	7.877E-12	2.84
5	0.025	2500	7.391E+01	7.735E+01	2.65	4.242E-11	1.973E-10	2.69

Tabla 8. Continuación.

Configuración			Michalewicz d=10			Shekel 7		
ns	Pr	It_{max}	Respuesta Promedio	Desviación Estándar	Tiempo Promedio [s]	Respuesta Promedio	Desviación Estándar	Tiempo Promedio [s]
1	0.005	12500	-9.63686	2.944E-02	6.29	-8.91841	2.780E+00	3.15
3	0.005	4166	-9.65515	1.210E-02	4.96	-8.52353	3.225E+00	2.53
5	0.005	2500	-9.65061	1.625E-02	4.77	-7.37620	3.572E+00	2.52
1	0.01	12500	-9.65348	1.408E-02	5.95	-9.87553	1.609E+00	3.21
3	0.01	4166	-9.65800	8.224E-03	4.96	-9.12844	2.621E+00	2.59
5	0.01	2500	-9.65966	1.499E-03	5.05	-7.65600	3.268E+00	2.50
1	0.015	12500	-9.65813	8.443E-03	6.15	-10.22714	9.629E-01	3.25
3	0.015	4166	-9.65966	1.499E-03	5.07	-10.00452	1.527E+00	2.62
5	0.015	2500	-9.65966	1.499E-03	4.85	-9.69835	1.827E+00	2.50
1	0.02	12500	-9.65950	1.698E-03	6.01	-10.40294	1.399E-15	3.13
3	0.02	4166	-9.65982	1.246E-03	5.03	-10.40294	9.330E-16	2.50
5	0.02	2500	-9.65982	1.246E-03	4.99	-9.52118	2.005E+00	2.51
1	0.025	12500	-9.65683	1.015E-02	6.09	-10.40294	1.475E-15	3.27
3	0.025	4166	-9.65982	1.246E-03	5.19	-10.40294	1.189E-15	2.63
5	0.025	2500	-9.65966	1.499E-03	5.14	-10.22714	9.629E-01	2.39
Configuración			Power Sum			Ackley		
ns	Pr	It_{max}	Respuesta Promedio	Desviación Estándar	Tiempo Promedio [s]	Respuesta Promedio	Desviación Estándar	Tiempo Promedio [s]
1	0.005	12500	6.633E-04	1.700E-03	5.03	1.984E-10	1.032E-09	3.82
3	0.005	4166	2.519E-03	3.833E-03	4.15	3.954E-11	4.594E-11	3.09
5	0.005	2500	2.670E-03	4.511E-03	4.00	1.283E-11	1.726E-11	2.96
1	0.01	12500	8.191E-04	2.133E-03	5.02	1.267E-10	1.442E-10	3.72
3	0.01	4166	2.137E-03	3.521E-03	4.14	4.470E-11	4.282E-11	3.19
5	0.01	2500	3.143E-03	3.709E-03	3.99	5.269E-11	8.502E-11	3.03
1	0.015	12500	4.193E-04	3.907E-04	4.99	2.007E-09	3.046E-09	3.93
3	0.015	4166	1.279E-03	3.316E-03	4.15	3.103E-10	1.288E-10	3.23
5	0.015	2500	1.145E-03	2.404E-03	3.99	4.354E-10	2.113E-10	3.08
1	0.02	12500	2.332E-04	2.506E-04	4.99	2.159E-08	3.214E-08	3.93
3	0.02	4166	1.218E-03	3.438E-03	4.14	1.018E-09	4.469E-10	3.23
5	0.02	2500	1.319E-03	3.200E-03	3.97	1.565E-09	2.011E-09	3.11
1	0.025	12500	1.524E-04	1.474E-04	4.96	3.388E-07	1.288E-07	3.99
3	0.025	4166	9.590E-04	1.824E-03	4.15	1.005E-07	8.038E-08	3.23
5	0.025	2500	1.410E-03	2.939E-03	3.98	1.205E-07	9.605E-08	3.02

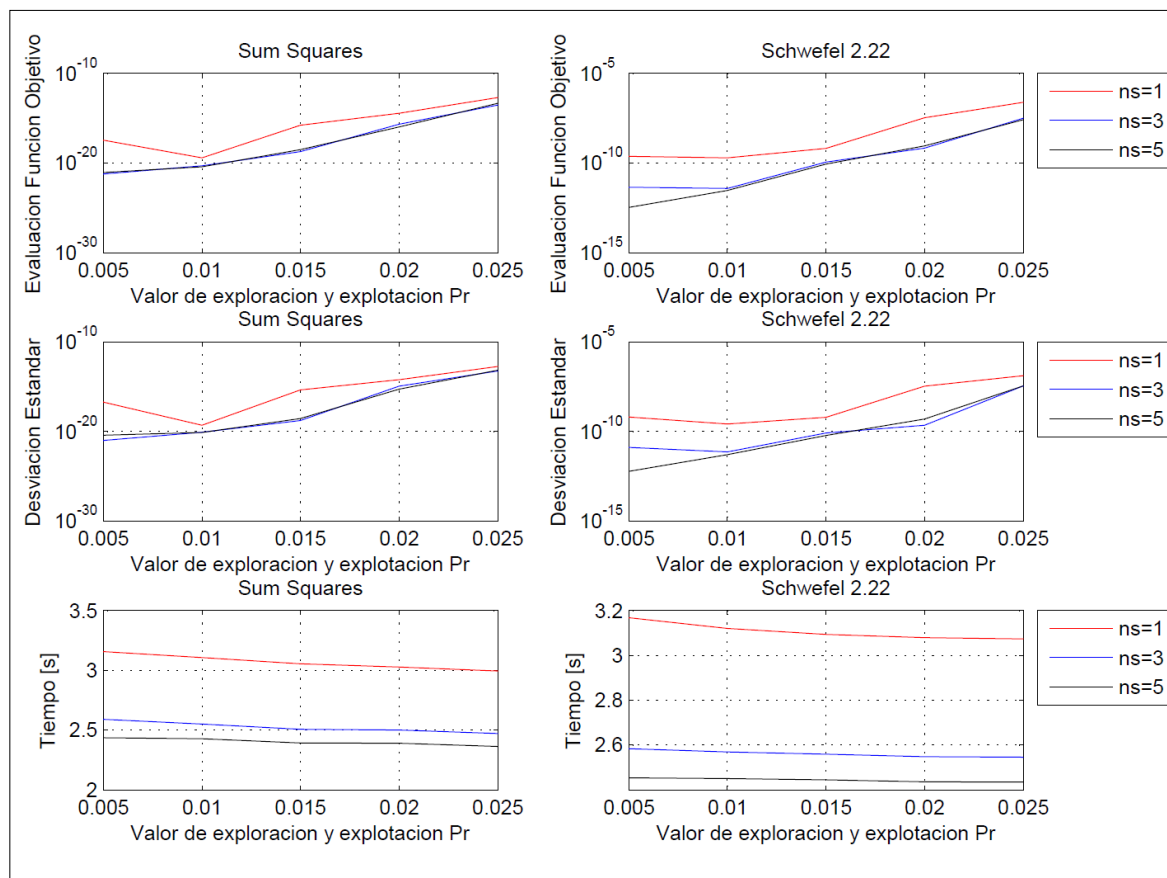


Figura 3.5 Resultados gráficos del estudio extensivo de parámetros para las funciones SumSquares y Schwefel 2.22.

Al analizar los resultados obtenidos en el estudio extensivo de parámetros, Tabla 8 y Figura 3.5, para la función Sum Squares, la cual es unimodal y separable; se observa que entre menor sea el valor de relación entre exploración y explotación Pr , la respuesta encontrada es mejor; aproximadamente del orden 1×10^{-20} para Pr igual a 0.005. En cambio cuando este parámetro aumenta, la media obtenida en la respuesta empieza a desmejorar; llegando al orden de 1×10^{-14} cuando Pr es igual a 0.025. Con respecto a la desviación estándar, se observa que describe un comportamiento similar al de la respuesta obtenida para el mínimo promedio; por lo tanto las conclusiones aplican para ambos casos. Se nota además que el valor ns no posee gran influencia en la respuesta; sin embargo debe encontrarse en el intervalo de 3 a 5 para no afectar la solución ni el tiempo promedio de ejecución del algoritmo, el cual es 0.5 segundos menor que cuando ns es igual a 1.

Realizando una reflexión similar para la función Schwefel 2.22, Figura 3.5, la cual es unimodal y no separable; se evidencia un comportamiento similar al ya observado en la

función SumSquares; debido a su característica unimodal. En el cual la función se comporta mejor con números bajos de Pr y un valor ns en el intervalo de 3 a 5.

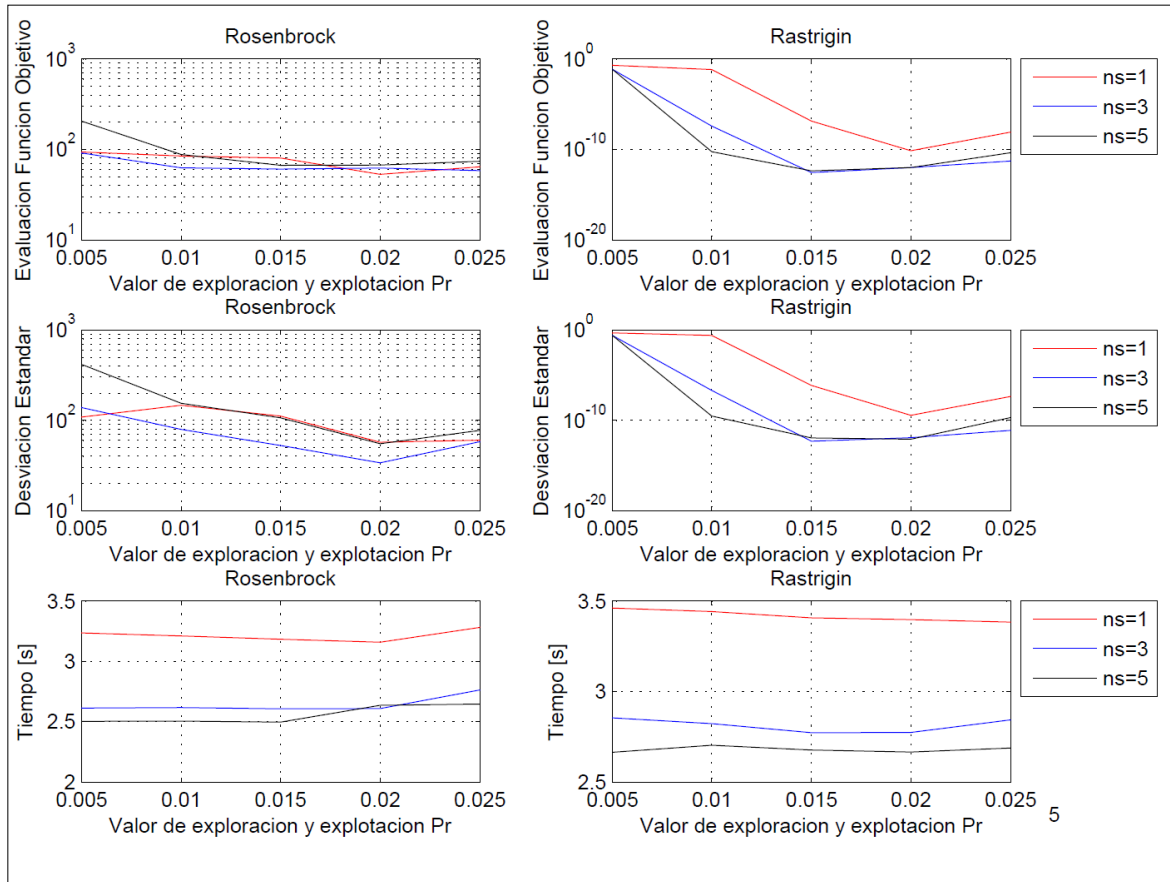


Figura 3.6 Resultados gráficos del estudio extensivo de parámetros para las funciones Rosenbrock y Rastrigin.

Al realizar el estudio en la función Rosenbrock, Figura 3.6, la cual es unimodal y no separable; se encuentra que un valor Pr bajo, el cual fomenta la explotación en el algoritmo, en este caso no presenta mejores resultados en calidad que uno medio o alto; comportamiento extraño en una función unimodal. Conducta ocasionada porque la función Rosenbrock, posee una propiedad que dificulta el seguimiento de los cambios de dirección; generando que la técnica creada necesite explorar en mayor medida el espacio de búsqueda, al no tener un patrón de desplazamiento claro para las ventosas del pulpo virtual. Por consiguiente un parámetro Pr dentro del intervalo 0.015 a 0.025, ofrece buenos resultados. Acerca del parámetro ns , se percibe que afecta la desviación estándar y el tiempo de ejecución; en el primer caso, es necesario que ns sea igual a 3 y para mantener un tiempo adecuado, debe ubicarse en el intervalo de 3 a 5.

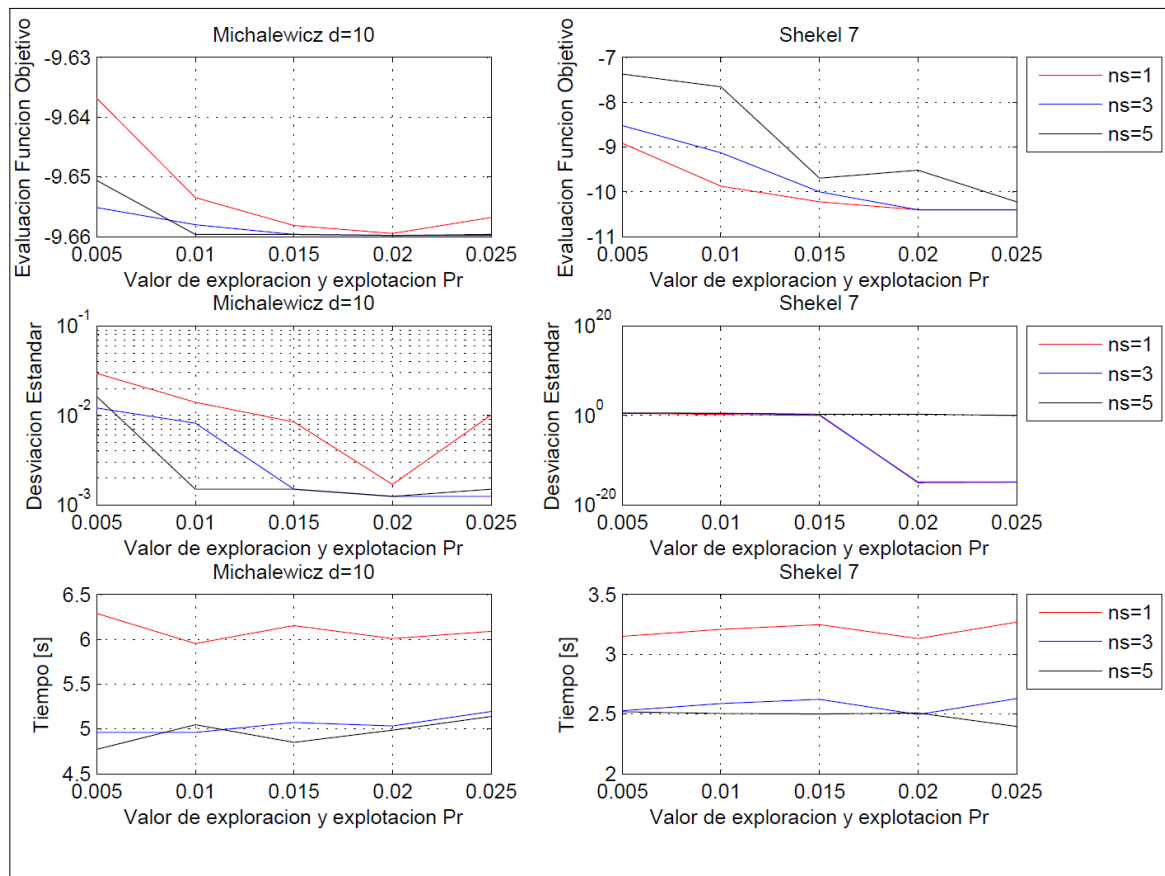


Figura 3.7 Resultados gráficos del estudio extensivo de parámetros para las funciones Michalewicz 10 y Shekel 7.

Las función Rastrigin que es multimodal y separable, Figura 3.6; requiere con la finalidad de obtener soluciones adecuadas, que el valor de Pr se encuentre en el intervalo de 0.015 a 0.025; ya que al ser una función con demasiados mínimos locales, necesita bastante exploración. Con respecto al parámetro ns , se advierte que una buena configuración de este, debe estar dentro del intervalo 3 a 5; ya que un valor de 1, hace que las ventosas existentes estén muy cercanas al pico virtual del animal, ocasionando que no halla la suficiente exploración. La desviación estándar en la función Rastrigin, sigue una conducta similar a la encontrada en su respuesta para el mínimo de la función objetivo; haciendo las conclusiones previas validas también para este caso. Con respecto al tiempo de ejecución del algoritmo, se puede mencionar lo que ya se ha observado en las anteriores funciones, que un valor ns bajo aumenta su duración.

La función Michalewicz 10 cuyo resultado se muestra en la Tabla 8, al igual que el problema Rastrigin es multimodal y separable; por ello presenta una sensibilidad similar al

parámetro Pr . Para que la solución sea cercana a la óptima, debe encontrarse en el intervalo de 0.01 a 0.025 siempre que ns este en el intervalo de 3 a 5; aunque si se observa la desviación estándar para esta función en la Figura 3.7, se concluye que el valor de Pr debe ser superior a 0.015. La sensibilidad del tiempo de ejecución es similar a las otras funciones analizadas, sufriendo poca influencia del parámetro Pr y una preferencia por valores medios o altos de ns .

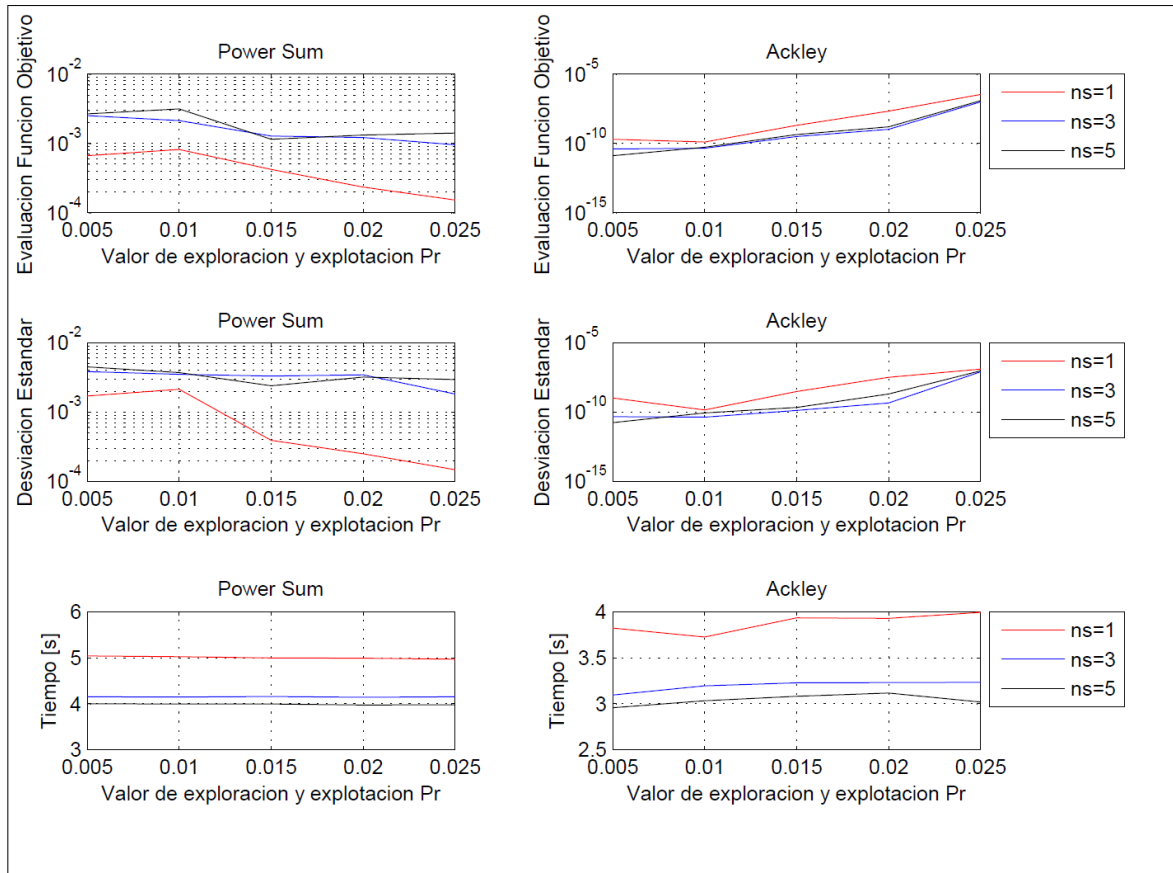


Figura 3.8 Resultados gráficos del estudio extensivo de parámetros para las funciones Power Sum y Ackley.

Funciones multimodales y no separables, como Shekel 7, Power Sum y Ackley, presentan comportamientos dispares entre sí; estas diferencias se describen a continuación:

Como se observa en la Tabla 8, la función Shekel 7, Figura 3.7, necesita para una respuesta cercana al óptimo, valores de Pr en el rango de 0.02 a 0.025 y valores de ns de 1 a 3. Si se observa el comportamiento de la desviación estándar, se corrobora la anterior conclusión.

Al mirar la gráfica del tiempo, se presenta un comportamiento similar al de funciones anteriores; por lo que la única opción conveniente para esta función, es un valor $ns = 3$.

Para el problema de prueba Power Sum, Figura 3.8, el algoritmo requiere, configuraciones de ns igual a 1 y cualquier valor de Pr en el rango de 0.02 a 0.025. La técnica metaheurística se desempeña bien al evaluar la función Ackley, cuando Pr está en el intervalo de 0.005 a 0.015, el cual es un comportamiento similar a los observados en las funciones unimodales, lo cual es extraño ya que esta función posee demasiados mínimos locales; sin embargo posee un parte convexa en la región central, lo que puede generar una necesidad mayor por explotación que por exploración.

Tabla 9. Mejores configuraciones de parámetros y límites de los parámetros ns y Pr .

Límites para cada parámetro				
Funciones	Intervalo parámetro ns	Intervalo parámetro Pr	Mejores Configuraciones	
			ns	Pr
Sum Squares	$3 \leq ns \leq 5$	$0.005 \leq Pr \leq 0.015$	3	0.005
Schwefel 2.22	$3 \leq ns \leq 5$	$0.005 \leq Pr \leq 0.015$	5	0.005
Rosenbrock	$ns = 3$	$0.015 \leq Pr \leq 0.025$	1	0.02
Rastrigin	$3 \leq ns \leq 5$	$0.015 \leq Pr \leq 0.025$	3	0.015
Michalewicz d=10	$3 \leq ns \leq 5$	$0.015 \leq Pr \leq 0.025$	5	0.02
Shekel 7	$ns = 3$	$0.02 \leq Pr \leq 0.025$	3	0.02
Power Sum	$ns = 1$	$0.02 \leq Pr \leq 0.025$	1	0.025
Ackley	$3 \leq ns \leq 5$	$0.005 \leq Pr \leq 0.015$	5	0.005
Promedio			3.25	0.0143

Al examinar los resultados en la Tabla 9, se nota que el parámetro $ns = 3$ sería una buena configuración para todas las funciones; a excepción del problema Power Sum. Notando que el detrimento ocasionado en esta función no es tan alto y beneficia en gran medida al resto de funciones; se selecciona este valor; el cual es bastante cercano al promedio para las mejores configuraciones de cada función ($ns = 3.25$); como se observa en la Tabla 9.

Con respecto a la configuración seleccionada para el parámetro Pr , se observa que según los resultados obtenidos en el estudio, el algoritmo encuentra solución en la mayoría de funciones, al seleccionar el parámetro Pr igual a 0.015; el cual es similar al promedio para las mejores configuraciones de cada función ($Pr = 0.0143$); ver Tabla 9. Sin embargo para funciones como shekel 7 y Power sum, el algoritmo demanda un valor mínimo de Pr igual

a 0.02; por lo tanto buscando mantener un funcionamiento equilibrado del algoritmo, se seleccionó Pr igual a 0.0175, el cual está en el intermedio de estos dos valores.

Los parámetros convenientes para la ejecución del algoritmo, son aquellos consignados en la Tabla 10.

Tabla 10. Configuración seleccionada de parámetros para la metaheurística AOOA.

Configuración seleccionada de parámetros	
Pr	0.0175
ns	3

Luego de haber analizado la sensibilidad en la respuesta, el tiempo y la desviación estándar de algunas funciones a los parámetros Pr y ns ; a continuación se sintetizan los efectos que producen estos parámetros en el algoritmo diseñado.

El número de ventosas por brazo ns , aparte de determinar directamente la población de la metaheurística; como se observa en la ecuación (2.3), influye principalmente en el tiempo de ejecución del algoritmo, ver Tabla 8; de forma que entre mayor sea este parámetro, menor será el tiempo de ejecución del algoritmo. Debido a que, aunque el número de evaluaciones de la función objetivo permanece constante, al aumentar ns la cantidad de iteraciones máximas It_{max} disminuye; ocasionando una reducción en los procesos realizados por el algoritmo y por lo tanto en el tiempo empleado en su ejecución. El otro efecto del parámetro ns , es la influencia en la exploración y explotación del algoritmo; ya que cuando es muy pequeño, las ventosas del animal explotan principalmente la posición del pico \hat{B} . En cambio al pasar el umbral de 3 ventosas por brazo, la exploración aumenta, al lograr que las ventosas estén más lejos del pico del octópodo virtual.

En cuanto al parámetro Pr , se puede mencionar que efectivamente realiza la función para la cual fue diseñado; controlar la proporción entre la exploración y explotación del algoritmo. Por ejemplo cuando el valor de este parámetro es bajo, el algoritmo realiza más explotación que exploración; comportamiento corroborado al analizar la respuesta de las funciones unimodales, SumSquares y Schwefel 2.22, Figura 3.5. En cambio cuando este parámetro se establece en números altos, el algoritmo explora en mayor medida el espacio de búsqueda; conducta ratificada por las soluciones encontradas para las funciones multimodales Rastrigin, Figura 3.6, Michalewicz 10 y Shekel 7, Figura 3.7.

4. Aplicación a un Problema de Ingeniería

Inicialmente se expone el problema de optimización seleccionado, describiendo claramente la función objetivo, sus variables y restricciones. Seguidamente se presentan los resultados obtenidos con la técnica AOOA, especificando los parámetros empleados para la ejecución. Luego se realiza un análisis de los resultados obtenidos y a la vez se determina que tal fue el desempeño de AOOA frente a los resultados obtenidos por otras técnicas de optimización metaheurística.

4.1 Problema de optimización de la viga soldada

Se debe optimizar el diseño de una viga soldada Figura 4.1, procurando minimizar el costo de su fabricación. El diseño está restringido por el esfuerzo cortante (τ), el esfuerzo de flexión (σ), la carga de pandeo (P_c), la deflexión final de la viga (δ) y las restricciones de las ecuaciones (4.3) a (4.9), [102].

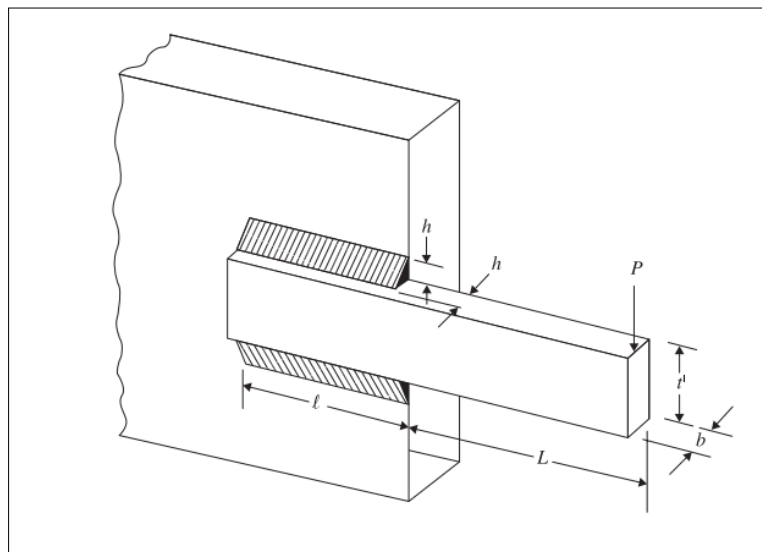


Figura 4.1 Esquema de la viga soldada. [102]

Las variables de optimización del problema son: la altura de la soldadura $h = x_1$, la longitud soldada de la barra $\ell = x_2$, la altura de la viga $t' = x_3$ y el espesor de la barra $b = x_4$.

Por lo tanto el vector de variables es:

$$\hat{\mathbf{x}} = [x_1, x_2, x_3, x_4] = [h, \ell, t', b] \quad (4.1)$$

En la ecuación (4.2) se especifica la función objetivo, la cual determina el costo de la viga.

$$f(\hat{\mathbf{x}}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \quad (4.2)$$

Sujeto a:

$$g_1(\hat{\mathbf{x}}) = \tau(\hat{\mathbf{x}}) - \tau_{max} \leq 0 \quad (4.3)$$

$$g_2(\hat{\mathbf{x}}) = \sigma(\hat{\mathbf{x}}) - \sigma_{max} \leq 0 \quad (4.4)$$

$$g_3(\hat{\mathbf{x}}) = x_1 - x_4 \leq 0 \quad (4.5)$$

$$g_4(\hat{\mathbf{x}}) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \leq 0 \quad (4.6)$$

$$g_5(\hat{\mathbf{x}}) = 0.125 - x_1 \leq 0 \quad (4.7)$$

$$g_6(\hat{\mathbf{x}}) = \delta(\hat{\mathbf{x}}) - \delta_{max} \leq 0 \quad (4.8)$$

$$g_7(\hat{\mathbf{x}}) = P - P_c(\hat{\mathbf{x}}) \leq 0 \quad (4.9)$$

Los límites superiores e inferiores de las variables son:

$$0.1 \leq x_1 \leq 2 \quad (4.10)$$

$$0.1 \leq x_2 \leq 10 \quad (4.11)$$

$$0.1 \leq x_3 \leq 10 \quad (4.12)$$

$$0.1 \leq x_4 \leq 2 \quad (4.13)$$

Donde:

$$\tau(\hat{\mathbf{x}}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \quad (4.14)$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2} \quad (4.15)$$

$$\tau'' = \frac{MR}{J} \quad (4.16)$$

$$M = P \left(L + \frac{x_2}{2} \right) \quad (4.17)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \quad (4.18)$$

$$J = 2 \left\{ \sqrt{2} x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2 \right] \right\} \quad (4.19)$$

$$\sigma(\hat{\mathbf{x}}) = \frac{6PL}{x_4 x_3^2} \quad (4.20)$$

$$\delta(\hat{\mathbf{x}}) = \frac{4PL^3}{E x_3^3 x_4} \quad (4.21)$$

$$P_c = \frac{4.013E \sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2} \left(1 - \frac{x_3}{2L} \sqrt{\frac{E}{4G}} \right) \quad (4.22)$$

Los parámetros del problema de optimización son los siguientes:

Carga en la viga $P = 6000 \text{ lb}$, longitud de la viga en voladizo $L = 14 \text{ in}$, deflexión máxima permitida al final de la viga $\delta_{max} = 0.25 \text{ in}$, módulo de elasticidad del material de la viga $E = 30 \times 10^6 \text{ psi}$, módulo de rigidez del material de la viga $G = 12 \times 10^6 \text{ psi}$, esfuerzo cortante máximo $\tau_{max} = 13600 \text{ psi}$, esfuerzo de flexión máximo $\sigma_{max} = 30000 \text{ psi}$.

4.2 Resultados con AOOA al problema de diseño de la viga soldada

El problema de la viga soldada fue solucionado por medio de AOOA con una cantidad de ventosas ns igual a 3 y un valor de 0.0175 para su parámetro Pr . Para el manejo de restricciones del problema; se empleó una función de penalización estática de tipo exterior, propuesta por Morales [103]. Esta función se muestra en la ecuación (4.23), donde K' es una constante, la cual para este caso es igual a 10^5 , m' es el número total de restricciones y s' es la cantidad de restricciones satisfechas por alguna solución $\hat{\mathbf{x}}$. Adicionalmente para delimitar el espacio de búsqueda dentro de los límites de las variables, se utilizó un método de pared absorbente.

$$f(\hat{\mathbf{x}}) = \begin{cases} f(\hat{\mathbf{x}}) & \text{Solucion es factible} \\ K' - \sum_{i=1}^{s'} \left(\frac{K'}{m'} \right) & \text{Solucion no factible} \end{cases} \quad (4.23)$$

Los valores óptimos para las variables de diseño encontradas por AOOA, se encuentran en la Tabla 11; allí se observa que para las condiciones de carga, el diseño de la viga óptima necesita una altura de la soldadura igual a 0.205724 in , una longitud soldada de 3.470494 in , la altura de la viga debe ser de 9.036878 in y debe tener un espesor de 0.205728 in ; valores que se encuentran dentro de los rangos establecidos para las variables del problema, ecuaciones (4.10) a (4.13).

Tabla 11. Resultados óptimos obtenidos con AOOA para el problema de la viga soldada.

Variables	h	ℓ	t'	b	Costo
Resultados óptimos con AOOA	0.205724	3.470494	9.036878	0.205728	1.724879

Tabla 12. Resultados obtenidos para las restricciones en el problema de la viga soldada.

g_1	g_2	g_3	g_4	g_5	g_6	g_7
2.141×10^{-5}	-1.502789	-3.535×10^{-6}	-3.432949	-0.080725	-0.235541	-3.431×10^{-5}

Según los valores presentados en la Tabla 12, se verifica que se satisfacen todas las restricciones impuestas en las ecuaciones (4.3) a (4.9); por lo tanto la solución encontrada es factible. Además se concluye de estos resultados, que el esfuerzo cortante, el esfuerzo de flexión, la carga de pandeo y la deflexión final de la viga, son admisibles según los criterios de diseño establecidos para este caso real de ingeniería. Lo cual permite concluir que AOOA es un algoritmo apto para la resolución de problemas reales de ingeniería.

4.3 AOOA frente a otras técnicas de optimización para el problema de diseño de la viga soldada

Los resultados encontrados por AOOA para el problema de diseño de la viga soldada, son expuestos en la Tabla 13 y Tabla 14, junto a los resultados encontrados por otras técnicas de optimización metaheurística y matemática. Los métodos RANDOM, SIMPLEX, DAVID y APPROX, son métodos matemáticos utilizados por Ragsdell [104]; el resto de algoritmos empleados son metaheurísticos. En el Anexo B, se encuentran los resultados de las 30 ejecuciones de AOOA al solucionar este problema de optimización.

En la Tabla 13 se observa que AOOA obtiene la tercera mejor solución para la función objetivo; ubicándose después de CBO e IHS, quienes están en primer y segundo lugar respectivamente. Sin embargo la diferencia con estos es mínima, siendo de 2.16×10^{-4} con CBO y de 7.9×10^{-5} con IHS.

En la Tabla 14 se presentan los resultados estadísticos obtenidos por varias técnicas de optimización, al resolver con 30 corridas independientes el problema de la viga soldada; allí se aprecia cómo AOOA posee el tercer lugar en mejor costo promedio con diferencias de 1.088×10^{-2} con el mejor resultado (CBO) y 4.583×10^{-3} con el segundo resultado (WOA). No obstante en los valores de desviación estándar, obtiene el segundo mejor resultado; lo que implica que AOOA es un algoritmo robusto en cuanto a la repetitividad de la respuesta.

Tabla 13 Comparación de los resultados de AOOA para el problema de optimización de la viga soldada.

Algoritmo	Variables óptimas				Costo óptimo
	h	ℓ	t'	b	
AOOA (Propuesto)	0.205724	3.470494	9.036878	0.205728	1.724879
WOA [11]	0.205396	3.484293	9.037426	0.206276	1.730499
GSA [11]	0.182129	3.856979	10.00000	0.202376	1.879952
CBO [105]	0.205722	3.47041	9.037276	0.205735	1.724663
RO [106]	0.203687	3.528467	9.004233	0.207241	1.735344
IHS [107]	0.20573	3.47049	9.03662	0.20573	1.7248
GA (Coello & Montes) [108]	0.205986	3.471328	9.020224	0.20648	1.728226
GA (Deb) [109]	0.2489	6.1730	8.1789	0.2533	2.4331
HS [110]	0.2442	6.2231	8.2915	0.2443	2.3807
RANDOM [104]	0.4575	4.7313	5.0853	0.6600	4.1185
SIMPLEX [104]	0.2792	5.6256	7.7512	0.2796	2.5307
DAVID [104]	0.2434	6.2552	8.2915	0.2444	2.3841
APPROX [104]	0.2444	6.2189	8.2915	0.2444	2.3815

Tabla 14 Resultados estadísticos de diferentes técnicas de optimización, para el problema de diseño de la viga soldada.

Algoritmo	Mejor resultado	Costo promedio	Peor resultado	Desviación estándar
AOOA (Propuesto)	1.724879	1.736583	1.787060	0.014653
WOA [11]	1.730499	1.7320	N/A	0.0226
GSA [11]	1.879952	3.5761	N/A	1.2874
CBO [105]	1.724663	1.725707	1.725059	0.0002437
GA (Coello & Montes) [108]	1.728226	1.792654	1.993408	0.074713

5. Conclusiones

5.1 Contribuciones

Diseño, desarrollo e implementación del algoritmo de optimización del pulpo artificial, la cual es una nueva técnica de optimización metaheurística para solucionar problemas de optimización mono-objetivo. Esta técnica se inspira en los pulpos, específicamente en la distribución espacial del animal, sus habilidades cognitivas, la dimensión longitudinal de sus extremidades, movimientos de sus brazos para alcanzar objetos, la habilidad de regeneración de miembros y órganos de los octópodos y por último en su locomoción.

A conocimiento del autor y en una búsqueda exhaustiva en la literatura, no se encontró ninguna técnica de optimización metaheurística inspirado en pulpos; por lo tanto se contribuye con la creación de la primera técnica metaheurística de optimización que simula a estos octópodos. Mostrando a la vez, el potencial que poseen estos cefalópodos como fuente de inspiración para la realización de técnicas de optimización.

La elaboración de un estudio de desempeño de AOOA con 42 problemas de prueba de la literatura; en el que se evaluaron las siguientes propiedades: la habilidad de este algoritmo para encontrar el óptimo de cada función, la destreza para escapar de mínimos locales y su comportamiento de convergencia.

La realización de dos estudios comparativos, el primero entre AOOA, GA, PSO, DE y ABC; el segundo entre, AOOA, CES, ESLAT, CMA-ES y ABC. Donde se verifico la capacidad de la técnica creada para resolver eficientemente problemas de optimización, con resultados competitivos frente a otras técnicas de optimización metaheurística.

Un estudio extensivo de parámetros de la técnica propuesta; en el cual se estudia la sensibilidad de la respuesta, la desviación estándar y el tiempo de ejecución del algoritmo, en función de sus dos parámetros ns y Pr ; en el que se establecen las bases para que los usuarios puedan seleccionar valores adecuados para estos parámetros.

La solución de un problema real de diseño mecánico con restricciones, por medio de AOOA, en el que el algoritmo encuentra resultados competitivos frente a otras técnicas de optimización; demostrando la aplicabilidad del algoritmo a casos reales de ingeniería.

5.2 Resultados

AOOA posee un desempeño destacable al hallar el mínimo en un amplio espectro de funciones de optimización; entregando excelentes resultados tanto en cercanía al óptimo y alta tasa de éxito de las soluciones halladas. Deducción obtenida al haberlo evaluado en funciones unimodales separables, unimodales no separables, multimodales separables y multimodales no separables. Donde AOOA encuentra el óptimo en el 71.42% de las 42 funciones de prueba consideradas; asegurando la repetitividad de estas soluciones, por los valores de desviación estándar obtenidos.

AOOA presenta baja repetitividad de buenos resultados en funciones con valle de curvatura estrecho, como lo son las funciones Rosenbrock, Colville y Dixon Price; en las cuales el óptimo global de la función, fue logrado solo en algunas ocasiones.

AOOA es un algoritmo con mínima intervención del usuario, puesto que además de la población y el número de evaluaciones permitidas de la función objetivo, que son parámetros intrínsecos de cualquier metaheurística; únicamente debe seleccionar dos parámetros, uno de ellos Pr , el cuál controla la relación entre explotación y exploración del algoritmo y ns que es el número de ventosas por brazo del octópodo virtual, el cual influencia directamente la población de la metaheurística y el tiempo de ejecución del algoritmo.

Al haber comparado el desempeño de la técnica diseñada en la solución de problemas, frente a otras técnicas de optimización metaheurística, se demostró que AOOA presenta resultados competitivos; es más, en algunos casos los resultados del algoritmo de optimización del pulpo artificial son sobresalientes, como sucede en las funciones Kowalik, Hartman 6, Fletcher-Powell 5, Perm, Power Sum y Fletcher-Powell 10; donde AOOA se desempeña mejor que los otros algoritmos aquí considerados. Verificando su validez como una nueva técnica de optimización metaheurística.

Al solucionar un problema de diseño mecánico con restricciones, AOOA arrojó resultados relevantes al ser comparado en un grupo de 13 técnicas, donde AOOA se ubica en tercer

lugar, después de CBO e IHS, quienes están en primer y segundo lugar respectivamente con diferencias de 2.16×10^{-4} con CBO y de 7.9×10^{-5} con IHS. Lo que permite concluir que el algoritmo diseñado, es efectivo para resolver problemas de optimización reales con restricciones.

5.3 Trabajos Futuros

El estudio realizado en este documento crea caminos para continuar avanzando en este campo de investigación; por lo tanto a continuación se definen claramente estos nuevos trabajos.

Técnica AOOA con sintonización automática de parámetros.

Un algoritmo del pulpo artificial para problemas de optimización multi-objetivo.

Un nuevo operador de exploración para AOOA, el cual mejore la repetitividad de buenos resultados en funciones con valle de curvatura estrecho. Este operador podría estar inspirado en otras características del animal o posiblemente en mayor cantidad de pulpos.

Una versión binaria de AOOA, para la resolución de problemas de tipo discreto.

Una comparación con otras técnicas de optimización metaheurística, utilizando el “Wilcoxon signed-rank test” [52]. Para determinar la posible superioridad del algoritmo creado.

Por último se propone la diversificación de AOOA, al utilizarlo en variados problemas reales de optimización.

Anexo A: Implementación del algoritmo en el lenguaje de MATLAB

En este anexo se presenta el código de AOOA implementado en el software MATLAB; este se compone de la función principal zAOOA, funciones secundarias zDobleArm, zEvalFPrueba, zFunHandler, zLimFObj, zMejorVentosa y zRegen. Se incluyen también cuatro funciones matemáticas de prueba, la función Sphere (unimodal y separable), la función Matyas (unimodal y no separable), la función Rastrigin (multimodal y separable) y por último la función Hartman 3 (multimodal no separable)

zAOOA

Esta es la función principal, la cual contiene la mayoría de operadores de la metaheurística AOOA. Las variables de entrada para su ejecución son: SFObj, es el número que identifica los problemas de prueba en la en la zFunHandler; ns, es el número de ventosas por brazo; Itmax, es el número máximo de iteraciones del algoritmo; Pr, es el valor de relación entre exploración y explotación, se recomienda seleccionar valores en entre 0.005 y 0.02; TGráficas, es una variable que determina si imprimir o no las gráficas según su valor, puede ser cero (0) para desactivar las gráficas o uno (1) para visualizar la curva de convergencia.

Los resultados de esta función son los siguientes: Min, es el valor mínimo encontrado al evaluar la función objetivo; HistoryBp, es el historial de la mejor posición global; Xmin, es la variable que contiene el vector de las variables de decisión óptimas encontradas.

Los comandos para correr las cuatro funciones disponibles para 4000 iteraciones del algoritmo, con ns igual a 3, Pr igual a 0.0175 e imprimiendo las curvas de convergencia, son:

```
>> [A, B, C]=zAOOA(1, 3, 4000, 0.0175, 1);  
>> [A, B, C]=zAOOA(2, 3, 4000, 0.0175, 1);  
>> [A, B, C]=zAOOA(3, 3, 4000, 0.0175, 1);  
>> [A, B, C]=zAOOA(4, 3, 4000, 0.0175, 1);
```

```

function [Min,HistoryBp,XMin]=zAOOA(SFObj,ns,Itmax,Pr,TGraficas)
%-----
% SFObj Funcion Objetivo a evaluar.
% ns Numero de ventosas por brazo, rango [1,2,...). Entero.
% Itmax Numero maximo de iteraciones.
% Pr Parametro de exploracion y explotacion, rango (0,0.1)...
% valores recomendados entre 0.005 y 0.02.
% TGraficas 0 desactivadas, 1 Curva de convergencia.
% XMin valores de las variables de decision optimas.
% HistoryBp El historico del valor global de la mejor posicion Bp,
f(Bp).% Min valor minimo que se obtiene en la funcion.
%Codigo de la metaheuristica AOOA.
% Autor: Eduardo Andres Gonzalez Guzman.
% Ultima actualizacion 08/03/2018.
%-----

%% DEFINE INTERVALOS A USAR Y SELECCION DE LA FUNCION DE PRUEBA
EFO=0;
HistoryBp=zeros(1,Itmax);
[DFunP,RFunP]=zLimFObj(SFObj);
[EFprueba,Label]=zFunHandler(DFunP(1,1));
close all;
rng('shuffle')
Iteration=1;

%% PARAMETROS y MATRICES
%-----Paso
inicial-----L=zeros(DFunP(1,2),ns,8);

for i=1:DFunP(1,2)
    for j=1:ns
        for k=1:8
            L(i,j,k)=rand*abs((RFunP(i,2))-(RFunP(i,1)))/(ns*8);
        end
    end
end

%-----Matrices Para almacenar Datos-----
S=zeros(DFunP(1,2),ns,8);% Ventosas
O=zeros(DFunP(1,2),8);% Direcciones prometedoras
Evafunob=zeros(8,ns,Itmax);

%% SECUENCIA DE ITERACIONES
while Iteration<=Itmax
% INICIAR EL PICO DEL ANIMAL B

if Iteration==1

```

```

    for h=1:DFunP(1,2)
        B(h,1)=RfunP(h,2)+(RfunP(h,1)-RfunP(h,2)).*rand();% B inicial
    end
end

EB=EFprueba(B);

%% INICIAR LAS VENTOSAS DEL ANIMAL

Sf=exp((-6/Itmax)*(Iteration-1));%Factor de encogimiento de tentáculos
Dir=zeros(8,2);%Para las direcciones prometedoras.
for jj=1:ns %ciclo según el número de ventosas por brazo

O=zDoblezArm(DFunP,O,Dir);%Orientación de la sección de brazo

    for kk=1:8 %Brazos
        if jj==1 || Dir(kk,2)==0
            S(:,jj,kk)=B+Sf*L(:,jj,kk).*O(:,kk)/sqrt(O(:,kk)'*O(:,kk));
        end

        if jj~=1 && Dir(kk,2)~=0
            if Dir(kk,1)==0
                S(:,jj,kk)=S(:,Dir(kk,2),kk)+Sf*L(:,jj,kk).*O(:,kk)/sqrt...
                    (O(:,kk)'*O(:,kk));
            else
                S(:,jj,kk)=S(:,Dir(kk,2),kk)+(1/(Pr*10)+rand)*Sf*L(:,jj,kk)...
                    .*O(:,kk)/sqrt(O(:,kk)'*O(:,kk));
            end
        end
    end

%% REGENERACION DE VENTOSAS

S=zRegen(Pr,RfunP,S,jj);% Regeneracion 1
if jj==ns % Regeneracion 2
    R3=randi(8);
    for Dim=1:DFunP(1,2)

        S(Dim,jj,R3)=RfunP(i,2)+(RfunP(i,1)-RfunP(i,2)).*rand();
    end
end

%% EVALUACION DE VENTOSAS, SELECCION DE VENTOSAS BSA Y DIRECCION O

Evafunob(:,jj,Iteration)=zEvalFPrueba(EFprueba,S,jj);% Evaluación Ventosas

if jj<=ns-1 %Selección Dirección Prometedora

```

```

    if jj==1
        for kk=1:8
            if Evafunob(kk,jj,Iteration)<=EB
                Dir(kk,1)=1;
                Dir(kk,2)=1;
            end
        end
    end

    if jj~=1
        for kk=1:8
            if Dir(kk,2)==0
                if Evafunob(kk,jj,Iteration)<=EB
                    Dir(kk,1)=1;
                    Dir(kk,2)=1;
                end
            end
        end

        if Dir(kk,2)~=0
            if Evafunob(kk,jj,Iteration)<=Evafunob(kk,Dir(kk,2),Iteration)
                Dir(kk,1)=1;
                Dir(kk,2)=Dir(kk,2)+1;
            else
                Dir(kk,1)=0;
            end
        end
    end
end

end

end

%% ACTUALIZACION DEL PASO L

for jj=1:ns
    for kk=1:8
        L(:,jj,kk)=abs(B-S(:,randi(ns),randi(8))));%Paso entre ventosas
    end
end

end

%% MATRIZ DE ACUMULACION DE MEJOR POSICION

BsI=zMejorVentosa(Evafunob,Iteration);% Indice de la mejor ventosa (Bs)
Bs=S(:,BsI(1,2),BsI(1,1));%Posicion de la mejor ventosa (Bs)

```

```

if Iteration==1
    Bp(:,1)=Bs; %Mejor Posicion de BP
    BpI(1,:)=BsI;
    BpI(1,3)=Iteration;
end
if Iteration~=1
if Evafunob(BsI(1,1),BsI(1,2),Iteration)<= Evafunob(BpI(1,1),BpI(1,2),BpI(1,3))
    Bp(:,1)=Bs;%Nueva posicion Mejor Posicion Bpde todas
    clear BpI
    BpI(1,:)=BsI;
    BpI(1,3)=Iteration;

end
end

%% NUEVO PICO DEL ANIMAL PO

if Iteration<=round(10*Pr*Itmax) %Simulacion Jetting

    B=Bs;% Mejor posicion de la Iteracion actual

        if rand<=Pr%Simulacion Jetting
            for h=1:DFunP(1,2)
                B(h,1)=RfunP(h,2)+(RfunP(h,1)-RfunP(h,2)).*rand();
                EFO=EFO+1;
            end
        end

else %luego de jetting

    if rand<=(1-Pr)

        B=Bp;% Mejor posicion de todas las Iteraciones
    else
        B=Bs;% Mejor posicion de la Iteracion actual
    end

end

%% PARA GRAFICAS

HistoryBp(1,Iteration)=EFprueba(Bp); % Historico mejores posiciones del pulpo

if T Graficas==1 % Graficas

```

```

    if Iteration==Itmax
    figure;
    plot(HistoryBp, 'LineWidth',2);
    semilogx(HistoryBp, 'LineWidth',2);
    xlabel('Iteración');
    ylabel('Mejor Evaluacion Funcion Objetivo');
    grid on;
    title(Label);
    print(strjoin(Label), '-dpng', '-r0')
    end
end

%% FIN DEL CICLO
Iteration=Iteration+1;

end % Termina el ciclo de iiteraciones

%% OBTENCION DEL MINIMO Y EFO
Min=EFprueba(Bp);% Arroja el minimo obtenido en la funcion objetivo.
XMin=Bp;% Arroja las mejores variables de decision para el minimo obtenido.
end

```

zDoblezArm

Función encargada de determinar las nuevas direcciones de orientación.

```

function [D] = zDoblezArm(DFunP,Usn,Dir)
    D=Usn;%Vector de direcciones por ventosa
    for kk=1:8
        if Dir(kk,1)==0
            D(:,kk)=(2*round(rand(DFunP(1,2),1))-1).*rand(DFunP(1,2),1);
        end
    end
end

```

zEvalFPrueba

Función que evalúa las ventosas del animal, contiene a la función de prueba que se ha seleccionado.

evaluar.

```

function [E] = zEvalFPrueba(EFprueba,A,jj)
Tam=size(A);
E=zeros(Tam(1,3),1);
    for ii=1:Tam(1,3)
        E(ii,1)=EFprueba(A(:,jj,ii));
    end
end

```

zMejorVentosa

Función que determina cual es la mejor ventosa en la iteración actual.

```
function [BSAll]=zMejorVentosa(Evafunob,iter)

Tam=size(Evafunob);

Bs=zeros(1,Tam(1,1));
BSAll=zeros(1,2);

for ii=1:Tam(1,1)
Bs(1,ii)=min(Evafunob(ii,:,iter));
end

[~, BSAll(1,1)]=min(Bs);
[~, BSAll(1,2)]=min(min(Evafunob(:, :, iter)));

end
```

zRegen

Función del operador Regeneración 1.

```
function [A] = zRegen(Pr,RFunP,A,jj)% regeneracion 1
Tam1=size(A);%dimension,sucker,arm

for ii=1:8%arm
for xx=1:Tam1(1,1)%dimension
if rand <=Pr
A(xx,jj,ii)=RFunP(xx,2)+(abs((RFunP(xx,2))-(RFunP(xx,1)))).*rand();
end
end
end
end
```

zFunHandler

Función que permite manejar las funciones de prueba; es decir las funciones a evaluar.

```
function [FH,Label] = zFunHandler(SFunP) % define las funciones de prueba

switch SFunP
    case 1 %Selecciona Sphere
        FH=@spheref;
        Label={'Funcion Sphere'};
    case 2 %Selecciona Matyas
        FH=@matya;
        Label={'Funcion Matyas'};
    case 3 %Selecciona Rastrigin
        FH=@rastr;
        Label={'Funcion Rastrigin'};
    case 4 %Selecciona Hartmann 3-D
        FH=@hart3;
        Label={'Funcion Hartman 3'};

end

end
```

zLimFObj

Función que controla el dominio del espacio de búsqueda y la cantidad de variables del problema.

```
function [A,B]=zLimFObj(SFObj)

switch SFObj
    case 1
        FunP=1;% Funcion Sphere d=1,2... (modf)
        dim=30;
        UL=100;
        LL=-100;
    case 2
        FunP=2;% Funcion Matyas d=1,2. (Nomodf)
        dim=2;
        UL=10;
        LL=-10;
    case 3
        FunP=3;% Funcion Rastrigin d=1,2... (modf)
        dim=30;
        UL=5.12;
        LL=-5.12;
```

```

    case 4
        FunP=4;% Funcion Hartmann 3-D d=1,2,3.(Nomodf)
        dim=3;
        UL=1;
        LL=0;
    end

    %FunP (Para zFunHandler; archivo con funciones Objetivo)
    %dim (Para dimension, cantidad de variables de la funcion de prueba)
    %en caso de nueva funcion colocarlas manualmente

    A=[FunP,dim];

    %Dominio

        B=zeros(dim,2);
        for x=1:dim
            B(x,:)=[UL,LL];
        end

    %En el caso de dominio diferente para cada variable,
    %crear matriz B con restricciones personalizadas.

        %B=[UL,LL;UL,LL;UL,LL];

    end

```

spheref

Función Sphere, monomodal y separable.

```

function [y] = spheref(xx)

d = length(xx);
sum = 0;
for ii = 1:d
    xi = xx(ii);
    sum = sum + xi^2;
end

y = sum;

end

```

matya

Función Matyas, monomodal y no separable.

```
function [y] = matya(xx)

x1 = xx(1);
x2 = xx(2);

term1 = 0.26 * (x1^2 + x2^2);
term2 = -0.48*x1*x2;

y = term1 + term2;

end
```

rastr

Función Rastrigin, multimodal y separable.

```
function [y] = rastr(xx)

d = length(xx);
sum = 0;
for ii = 1:d
    xi = xx(ii);
    sum = sum + (xi^2 - 10*cos(2*pi*xi));
end

y = 10*d + sum;

end
```

hart3

Función Hartman 3, de tipo multimodal no separable.

```
function [y] = hart3(xx)

alpha = [1.0, 1.2, 3.0, 3.2]';
A = [3.0, 10, 30;
     0.1, 10, 35;
     3.0, 10, 30;
     0.1, 10, 35];
P = [0.3689, 0.1170, 0.2673;
     0.4699, 0.4387, 0.7470;
     0.1091, 0.8732, 0.5547;
     0.03815, 0.5743, 0.8828];

outer = 0;
for ii = 1:4
    inner = 0;
    for jj = 1:3
        xj = xx(jj);
        Aij = A(ii, jj);
        Pij = P(ii, jj);
        inner = inner + Aij*(xj-Pij)^2;
    end
    new = alpha(ii) * exp(-inner);
    outer = outer + new;
end

y = -outer;

end
```

Anexo B: Resultados de las 30 ejecuciones del algoritmo en el problema de aplicación del diseño de la viga soldada

Costo	$h(x_1)$	$\ell(x_2)$	$t'(x_3)$	$b(x_4)$
1.724933	0.205670	3.471775	9.036624	0.205730
1.727511	0.203799	3.512536	9.036624	0.205730
1.731063	0.201316	3.568081	9.036624	0.205730
1.752467	0.188274	3.890267	9.036624	0.205730
1.763410	0.182607	4.048451	9.036624	0.205730
1.787061	0.223155	3.259399	8.676630	0.223155
1.734413	0.208392	3.436114	8.978719	0.208392
1.758361	0.203840	3.375648	9.403956	0.203966
1.725748	0.205074	3.484630	9.036666	0.205729
1.732628	0.200254	3.592351	9.036624	0.205730
1.725230	0.205451	3.476492	9.036625	0.205730
1.724880	0.205725	3.470494	9.036878	0.205728
1.756319	0.186214	3.946368	9.036624	0.205730
1.724923	0.205678	3.471608	9.036624	0.205730
1.742688	0.193867	3.745456	9.036624	0.205730
1.735162	0.198687	3.627950	9.038667	0.205719
1.734491	0.199017	3.621051	9.036626	0.205730
1.725179	0.205489	3.475679	9.036624	0.205730
1.747153	0.211948	3.391451	8.903070	0.211948
1.730883	0.201469	3.564432	9.037120	0.205727
1.728305	0.203234	3.525011	9.036624	0.205730
1.725270	0.205846	3.468971	9.034071	0.205846
1.729179	0.202619	3.538714	9.036625	0.205730
1.729168	0.202627	3.538538	9.036624	0.205730
1.730553	0.201666	3.560152	9.036626	0.205730
1.726191	0.204750	3.491687	9.036640	0.205730
1.747322	0.191149	3.814502	9.036619	0.205730
1.740491	0.210087	3.414646	8.942416	0.210087
1.725273	0.205420	3.477170	9.036624	0.205730
1.731260	0.201181	3.571149	9.036624	0.205730

Bibliografía

- [1] a Astolfi, “Optimization An introduction.,” *J. Anal. Psychol.*, vol. 55, no. 5, pp. 617–635, 2010.
- [2] X. S. Yang, “Optimization and Metaheuristic Algorithms in Engineering,” *Metaheuristics Water, Geotech. Transp. Eng.*, no. 2013, pp. 1–23, 2013.
- [3] J. Durillo, “Metaheuristics for Multi-objective Optimization : Design , Analysis , and Applications,” 2011.
- [4] R. T. Rockafellar, “Fundamentals of Optimization,” *Univ. Washingt.*, 2007.
- [5] J. Melorose, R. Perroy, and S. Careas, *HANDBOOK OF METAHEURISTICS*, vol. 1. 2015.
- [6] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: overview and conceptual comparison,” *ACM Comput. Surv.*, vol. 35, no. 3, pp. 189–213, 2003.
- [7] S. Voß, “Meta-heuristics : The State of the Art,” *Local Search Plan. Sched.*, pp. 1–23, 2001.
- [8] S. Saremi, S. Mirjalili, and A. Lewis, “Grasshopper Optimisation Algorithm: Theory and application,” *Adv. Eng. Softw.*, vol. 105, pp. 30–47, 2017.
- [9] X. S. Yang, S. Deb, and S. Fong, “Metaheuristic algorithms: Optimal balance of intensification and diversification,” *Appl. Math. Inf. Sci.*, vol. 8, no. 3, pp. 977–983, 2014.
- [10] O. Olorunda and A. P. Engelbrecht, “Measuring exploration/exploitation in particle swarms using swarm diversity,” in *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, 2008.

-
- [11] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, 2016.
- [12] Z. Beheshti and S. M. H. Shamsuddin, "A review of population-based meta-heuristic algorithm," *Int. J. Adv. Soft Comput. its Appl.*, vol. 5, no. 1, pp. 1–35, 2013.
- [13] B. Alatas, "ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 13170–13180, 2011.
- [14] K. C. B. Steer, A. Wirth, and S. K. Halgamuge, "The rationale behind seeking inspiration from nature," *Studies in Computational Intelligence*, vol. 193, pp. 51–76, 2009.
- [15] et al. Varela Iglesias J L, Tadeo Monge FT, Carreño Rujillo G, Doménech Colón F, Abad León A, "Real Academia Española. Diccionario Usual," *Edición del Tricentenario*. p. 1, 2014.
- [16] P. E. Services, M. Abo-zahhad, S. M. Ahmed, N. Sabor, J. Jia, X. Wang, J. Chen, F. V. C. Martins, E. G. Carrano, E. F. Wanner, R. H. C. Takahashi, G. R. Mateus, A. Norouzi, C. a C. Coello, G. B. Lamont, D. a. Van Veldhuizen, M. Rovcanin, E. De Poorter, D. van den Akker, I. Moerman, P. Demeester, C. Blondia, N. Zhu, I. O'Connor, S. Özdemir, B. a. Attea, Ö. a. Khalil, J. M. Lanza-Gutierrez, J. a. Gomez-Pulido, M. a. Vega-Rodriguez, J. M. Sanchez-Perez, L. Carlos, F. Carvalho, G. Engineers, T. Review, Y. Lakhdar, and E. H. Sbai, *Multi-objective evolutionary algorithms for energy-efficiency in heterogeneous wireless sensor networks*, vol. 11, no. 3. 2014.
- [17] M. Guzman, "Técnicas de optimización basadas em quimiotaxia de bacterias," 2009.
- [18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1989.
- [19] E. Alba, *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley and Sons, 2005.
- [20] B. Jarraya and A. Bouri, "Metaheuristic Optimization Backgrounds : A Literature Review," *Int J. Contemp. Bus. Stud.*, vol. 3, no. December, pp. 31–44, 2012.

-
- [21] F. Luna, “Metaheurísticas avanzadas para problemas reales en redes de telecomunicaciones,” pp. 19–38, 2008.
- [22] Á. García Sánchez, “Técnicas metaheurísticas,” p. 47.
- [23] A. Lazar and R. G. Reynolds, “Heuristic Knowledge Discovery for Archaeological Data Using Genetic Algorithms and Rough Sets,” *Artificial Intel. Lab. Dep. Comput. Sci. Wayne State Univ.*, no. September 2016, pp. 1–26, 2003.
- [24] K. Sörensen and S. Kenneth, “Metaheuristics – the Metaphor Exposed,” *Int. Trans. Oper. Res.*, vol. 22, no. September, pp. 1–20, 2012.
- [25] J. H. Holland, *Adaptation in Natural and Artificial Systems*, vol. Ann Arbor. 1975.
- [26] Z. Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs,” *Computational Statistics & Data Analysis*, vol. 24, no. 3. pp. 372–373, 1996.
- [27] C. Y. Ho, F. Y. Wang, C. C. Tseng, and Y. D. Lin, “NAT-compatibility testbed: An environment to automatically verify direct connection rate,” *IEEE Commun. Lett.*, vol. 15, no. 1, pp. 4–6, Jan. 2011.
- [28] M. Dorigo and G. Di Caro, “Ant colony optimization: a new meta-heuristic,” *Proc. 1999 Congr. Evol. Comput. (Cat. No. 99TH8406)*, vol. 2, pp. 1470–1477, 1999.
- [29] C. Blum, “Ant colony optimization: Introduction and recent trends,” *Physics of Life Reviews*, vol. 2, no. 4. pp. 353–373, Dec-2005.
- [30] M. Dorigo and C. Blum, “Ant colony optimization theory: A survey,” *Theor. Comput. Sci.*, vol. 344, no. 2–3, pp. 243–278, Nov. 2005.
- [31] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, *Metaheuristic Algorithms in Modeling and Optimization*, First Edit. Elsevier Inc., 2013.
- [32] J. Kennedy and R. Eberhart, “Particle swarm optimization,” *Neural Networks, 1995. Proceedings., IEEE Int. Conf.*, vol. 4, pp. 1942–1948 vol.4, 1995.
- [33] R. Storn and K. Price, “Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces,” *J. Glob. Optim.*, pp. 341–359, 1997.

-
- [34] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *Control Syst. IEEE*, vol. 22, no. 3, pp. 52–67, 2002.
- [35] H. Bremermann, "Chemotaxis and optimization," *J. Franklin Inst.*, vol. 297, no. 5, pp. 397–404, 1974.
- [36] B. H. H. Ocaña, E. Mezura-Montes, M. C. G. C. G. de los Santos Torres, and M. C. O. A. C. Bosquez, "Adaptación Del Forrajeo De Bacterias Para Resolver Problemas De Optimización Con Restricciones," *Lania.Mx*, 2009.
- [37] D. Karaboga, "An Idea Based On Honey Bee Swarm For Numerical Optimization," *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2013.
- [38] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Glob. Optim.*, vol. 39, no. 3, pp. 459–471, 2007.
- [39] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A Comprehensive Review of Swarm Optimization Algorithms," *PLoS One*, vol. 10, no. 5, p. e0122827, May 2015.
- [40] F. S. Abu-Mouti and M. E. El-Hawary, "Overview of Artificial Bee Colony (ABC) algorithm and its applications," in *SysCon 2012 - 2012 IEEE International Systems Conference, Proceedings*, 2012.
- [41] X. Yang, *Nature-Inspired Metaheuristic Algorithms Second Edition*. 2010.
- [42] M. K. Sayadi, R. Ramezani, and N. Ghaffari-Nasab, "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems," *Int. J. Ind. Eng. Comput.*, vol. 1, no. 1, pp. 1–10, Jun. 2010.
- [43] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems," *CAD Comput. Aided Des.*, vol. 43, no. 3, pp. 303–315, 2011.
- [44] A. E. M. Zavala, A. H. Aguirre, and E. R. Villa Diharce, "Particle evolutionary swarm optimization algorithm (PESO)," in *Proceedings of the Mexican International Conference on Computer Science*, 2005, vol. 2005, pp. 282–289.

-
- [45] W. Gong, Z. Cai, and D. Liang, "Cultured Differential Evolution for Constrained Optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 716–727, 2015.
- [46] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 17, no. 12, pp. 4831–4845, 2012.
- [47] H. Eskandar, A. Sadollah, A. Bahreininejad, and M. Hamdi, "Water cycle algorithm - A novel metaheuristic optimization method for solving constrained engineering optimization problems," *Comput. Struct.*, vol. 110–111, pp. 151–166, 2012.
- [48] Y. Gheraibia and A. Moussaoui, "Penguins Search Optimization Algorithm (PeSOA) BT - Recent Trends in Applied Artificial Intelligence: 26th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2013, Amsterdam, The Netherlands," M. Ali, T. Bosse, K. V Hindriks, M. Hoogendoorn, C. M. Jonker, and J. Treur, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 222–231.
- [49] S. Salcedo-Sanz, J. Del Ser, I. Landa-Torres, S. Gil-Lpez, and J. A. Portilla-Figueras, "The Coral Reefs Optimization Algorithm: A Novel Metaheuristic for Efficiently Solving Optimization Problems," *Sci. World J.*, vol. 2014, 2014.
- [50] M. Y. Cheng and D. Prayogo, "Symbiotic Organisms Search: A new metaheuristic optimization algorithm," *Comput. Struct.*, vol. 139, pp. 98–112, 2014.
- [51] S. A. Uymaz, G. Tezel, and E. Yel, "Artificial algae algorithm (AAA) for nonlinear global optimization," *Appl. Soft Comput. J.*, vol. 31, pp. 153–171, 2015.
- [52] J. F. Hilton, "The appropriateness of the Wilcoxon test in ordinal data," *Stat. Med.*, vol. 15, no. 6, pp. 631–645, 1996.
- [53] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1155–1173, 2008.
- [54] R. Villanueva, C. Nozais, and S. V. Boletzky, "Swimming behaviour and food searching in planktonic *Octopus vulgaris* Cuvier from hatching to settlement," *J. Exp. Mar. Bio. Ecol.*, vol. 208, no. 1–2, pp. 169–184, 1997.

-
- [55] J. Mather and D. Scheel, "Behaviour," in *Cephalopod Culture*, J. Iglesias, L. Fuentes, and R. Villanueva, Eds. Dordrecht: Springer Netherlands, 2014, pp. 17–39.
- [56] B. Hochner, "Octopuses," *Curr. Biol.*, vol. 18, no. 19, pp. R897–R898, Apr. 2017.
- [57] J. A. Mather, "Behaviour Development: A Cephalopod Perspective," *Int. J. Comp. Psychol.*, vol. 19, no. 1, 2006.
- [58] B. Mazzolai, L. Margheri, P. Dario, and C. Laschi, "Measurements of octopus arm elongation: Evidence of differences by body size and gender," *J. Exp. Mar. Bio. Ecol.*, vol. 447, pp. 160–164, 2013.
- [59] M. Barbato, M. Bernard, L. Borrelli, and G. Fiorito, "Body patterns in cephalopods. 'Polyphenism' as a way of information exchange," *Pattern Recognit. Lett.*, vol. 28, no. 14, pp. 1854–1864, 2007.
- [60] R. L. Caldwell, R. Ross, A. Rodaniche, and C. L. Huffard, "Behavior and body patterns of the Larger Pacific Striped Octopus," *PLoS One*, vol. 10, no. 8, pp. 1–17, 2015.
- [61] P. Central, "The Octopus : A Model for a Comparative Analysis of the Evolution of Learning and Memory Mechanisms," no. June, pp. 308–317, 2006.
- [62] I. Zarrella, G. Ponte, E. Baldascino, and G. Fiorito, "Learning and memory in *Octopus vulgaris*: a case of biological plasticity," *Curr. Opin. Neurobiol.*, vol. 35, no. i, pp. 74–79, 2015.
- [63] J. A. Mather and M. J. Kuba, "The cephalopod specialties: complex nervous system, learning, and cognition 1," *Can. J. Zool.*, vol. 91, no. 6, pp. 431–449, 2013.
- [64] J. A. Mather, "Cephalopod consciousness: Behavioural evidence," *Conscious. Cogn.*, vol. 17, no. 1, pp. 37–48, 2008.
- [65] B.-U. Budelmann, "Octopus. physiology and behaviour of an advanced invertebrate," *Behav. Processes*, vol. 3, no. 4, pp. 356–357, 1978.
- [66] M. J. Sweeney, C. F. E. Roper, K. M. Mangold, M. R. Clark, and S. V. Boletzky, "Larval and juvenile cephalopods: a manual for their identification," *Smithson. Contrib. to Zool.*, 1992.

-
- [67] F. Tramacere, L. Beccai, M. J. Kuba, and B. Mazzolai, "Octopus Suckers Identification Code (OSIC)," no. Voight 1991, 2013.
- [68] J. Z. YOUNG, "THE NUMBER AND SIZES OF NERVE CELLS IN OCTOPUS," *Proc. Zool. Soc. London*, 1963.
- [69] T. Shomrat, I. Zarrella, G. Fiorito, and B. Hochner, "The Octopus Vertical Lobe Modulates Short-Term Learning Rate and Uses LTP to Acquire Long-Term Memory," *Curr. Biol.*, vol. 18, no. 5, pp. 337–342, 2008.
- [70] C. Alves, J. G. Boal, and L. Dickel, "Short-distance navigation in cephalopods: A review and synthesis," *Cogn. Process.*, vol. 9, no. 4, pp. 239–247, 2008.
- [71] W. M. Kier and K. K. Smith, "The biomechanics of movement in tongues and tentacles," *Journal of Biomechanics*, vol. 16, no. 4. pp. 292–293, 1983.
- [72] "Kathleen K. Smith-Trunks, Tongues, and Tentacles Moving with Skeletons of Muscle (1989).pdf." .
- [73] L. Margheri, B. Mazzolai, M. Cianchetti, P. Dario, and C. Laschi, "Tools and methods for experimental in-vivo measurement and biomechanical characterization of an Octopus vulgaris arm," *Proc. 31st Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. Eng. Futur. Biomed. EMBC 2009*, pp. 7196–7199, 2009.
- [74] C. L. Huffard, R. L. Caldwell, and F. Boneka, "Mating behavior of *Abdopus aculeatus* (d'Orbigny 1834) (Cephalopoda: Octopodidae) in the wild," *Mar. Biol.*, 2008.
- [75] Y. Yekutieli, R. Mitelman, B. Hochner, and T. Flash, "Analyzing Octopus Movements Using Three-Dimensional Reconstruction," *J Neurophysiol*, vol. 98, no. 3, pp. 1775–1790, 2007.
- [76] Y. Yekutieli, R. Sagiv-Zohar, R. Aharonov, Y. Engel, B. Hochner, and T. Flash, "A dynamic model of the Octopus arm. I. Biomechanics of the Octopus reaching movement.," *J. Neurophysiol.*, vol. 5, pp. 291–323, 2005.
- [77] Y. Yekutieli, "Dynamic Model of the Octopus Arm. II. Control of Reaching Movements," *J. Neurophysiol.*, vol. 94, no. 2, pp. 1459–1468, 2005.

- [78] G. Sumbre, G. Fiorito, T. Flash, and B. Hochner, "Octopuses Use a Human-like Strategy to Control Precise Point-to-Point Arm Movements," *Curr. Biol.*, vol. 16, no. 8, pp. 767–772, 2006.
- [79] J. P. F eral, "Wound healing after arm amputation in *Sepia officinalis* (Cephalopoda: Sepioidea)," *J. Invertebr. Pathol.*, 1988.
- [80] J. R. Voight, "Movement, injuries and growth of members of a natural population of the Pacific pygmy octopus, *Octopus digueti*," *J. Zool.*, 1992.
- [81] S. M. Fossati, F. Carella, G. De Vico, F. Benfenati, and L. Zullo, "Octopus arm regeneration: Role of acetylcholinesterase during morphological modification," *J. Exp. Mar. Bio. Ecol.*, vol. 447, pp. 93–99, 2013.
- [82] C. L. Huffard, "Locomotion by *Abdopus aculeatus* (Cephalopoda: Octopodidae): walking the line between primary and secondary defenses.," *J. Exp. Biol.*, vol. 209, no. Pt 19, pp. 3697–3707, 2006.
- [83] G. E. G. Westermann, "Ammonoid life and habitat," in *Ammonoid paleobiology*, 1996.
- [84] M. J. Wells, "Oxygen extraction and jet propulsion in cephalopods," *Can. J. Zool.*, 1990.
- [85] J. A. Mather, "Navigation by spatial memory and use of visual landmarks in octopuses," *J. Comp. Physiol. A*, vol. 168, no. 4, pp. 491–497, 1991.
- [86] J. W. Forsythe and R. T. Hanlon, "Foraging and associated behavior by *Octopus cyanea* Gray, 1849 on a coral atoll, French Polynesia," *J. Exp. Mar. Bio. Ecol.*, vol. 209, no. 1–2, pp. 15–31, 1997.
- [87] M. J. Wells, G. G. Duthie, D. F. Houlihan, P. J. S. Smith, and J. Wells, "Blood Flow and Pressure Changes in Exercising Octopuses (*Octopus Vulgaris*)," *J. Exp. Biol.*, 1987.
- [88] J. A. Mather, "How Do Octopuses Use Their Arms?," *J. Comp. Psychol.*, 1998.
- [89] C. L. Huffard, F. Boneka, and R. J. Full, "Underwater Bipedal Locomotion by

- Octopuses in Disguise,” *Science (80-.)*, vol. 1, no. 4, pp. 1927–1927, 1927.
- [90] M. Jamil and X.-S. Yang, “A Literature Survey of Benchmark Functions For Global Optimization Problems Citation details: Momin Jamil and Xin-She Yang, A literature survey of benchmark functions for global optimization problems,” *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.
- [91] D. Karaboga and B. Akay, “A comparative study of Artificial Bee Colony algorithm,” *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132, 2009.
- [92] X. Yang, *Studies in Computational Intelligence 516 Cuckoo Search and Firefly Algorithm*. 2013.
- [93] D. Ortiz-Boyer, C. Hervás-Martínez, and N. García-Pedrajas, “A crossover operator for evolutionary algorithms based on population features,” *J. Artif. Intell. Res.*, vol. 24, pp. 1–48, 2005.
- [94] Z. Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs,” *Computational Statistics & Data Analysis*. 1996.
- [95] A. Hedar and M. Fukushima, “Evolution Strategies Learned with Automatic Termination Criteria,” *SCIS ISIS*, vol. 2006, pp. 1126–1134, 2006.
- [96] M. Birattari, *Tuning Metaheuristics*, vol. 197. 2009.
- [97] S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil, “Using experimental design to find effective parameter settings for heuristics,” *J. Heuristics*, vol. 7, no. 1, pp. 77–97, 2001.
- [98] J. N. Hooker, “Testing heuristics: We have it all wrong,” *J. Heuristics*, vol. 1, no. 1, pp. 33–42, 1995.
- [99] J. Xu and J. P. Kelly, “A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem,” *Transp. Sci.*, 1996.
- [100] B. Adenso-Díaz and M. Laguna, “Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search,” *Oper. Res.*, vol. 54, no. 1, pp. 99–114, 2006.

-
- [101] F. Hutter, F. Hutter, H. H. Hoos, and H. H. Hoos, "Automatic Algorithm Configuration based on Local Search," *BMC Bioinformatics*, 2007.
- [102] S. S. Rao, *Engineering Optimization: Theory and Practice*. 2009.
- [103] A. F. Kuri Morales and C. C. Quezada, "A universal eclectic genetic algorithm for constrained optimization," *Proc. 6th Eur. Congr. Intell. Tech. Soft Comput. EUFIT'98*, pp. 2–6, 1998.
- [104] K. M. Ragsdell and D. T. Phillips, "Optimal Design of a Class of Welded Structures Using Geometric Programming," *J. Eng. Ind.*, vol. 98, no. 3, pp. 1021–1025, 1976.
- [105] A. Kaveh and V. R. Mahdavi, "Colliding bodies optimization: A novel meta-heuristic method," *Comput. Struct.*, vol. 139, pp. 18–27, 2014.
- [106] A. Kaveh and M. Khayatazad, "A new meta-heuristic method: Ray Optimization," *Comput. Struct.*, vol. 112–113, pp. 283–294, 2012.
- [107] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Appl. Math. Comput.*, vol. 188, no. 2, pp. 1567–1579, 2007.
- [108] C. A. Coello Coello and E. M. Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," *Adv. Eng. Informatics*, vol. 16, no. 3, pp. 193–203, 2002.
- [109] K. Deb, "Optimal Design of a Welded Beam via Genetic Algorithms," *AIAA J.*, vol. 29, no. 11, pp. 2013–2015, 1991.
- [110] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice," *Comput. Methods Appl. Mech. Eng.*, vol. 194, no. 36–38, pp. 3902–3933, 2005.