Algoritmos Bioinspirados   
Artificial Bee Colony (ABC)

Junnior Santiago Ravelo

**Estudiante**

Henry David Suarez Serrano  
**Estudiante**

Javier Andrés Villarreal Amaya  
**Estudiante**

Juan Camilo Arciniegas  
**Estudiante**

José Gerardo Chacón Rangel

**Docente**

Universidad de Pamplona

2024-1

**Imagen del codigo**



**Codigo Escrito : Python**

import random

import math

# Definir la función objetivo

def objective\_function(x):

return x\*4 - 3\*x3 + 2\*x\*2 - x

# Clase para representar a una abeja

class Bee:

def \_init\_(self, min\_val, max\_val):

# Inicialización de la posición de la abeja de manera aleatoria dentro del rango dado

self.position = random.uniform(min\_val, max\_val)

# Cálculo del costo de la posición inicial

self.cost = objective\_function(self.position)

# Función principal del algoritmo de abejas

def bee\_algorithm(min\_val, max\_val, num\_bees, max\_epochs):

# Crear un conjunto de abejas con posiciones aleatorias

bees = [Bee(min\_val, max\_val) for \_ in range(num\_bees)]

# Iterar sobre las épocas (iteraciones) del algoritmo

for epoch in range(max\_epochs):

# Etapa de exploración de abejas empleadas

for i in range(num\_bees):

# Generar una posición candidata cercana a la posición actual

new\_position = bees[i].position + random.uniform(-8, 10) \* (random.choice([-8, 10]) \* 0.08 \* max\_val)

# Asegurar que la nueva posición esté dentro de los límites

new\_position = max(min\_val, min(new\_position, max\_val))

# Calcular el costo de la nueva posición

new\_cost = objective\_function(new\_position)

# Actualizar la posición si la nueva posición es mejor

if new\_cost < bees[i].cost:

bees[i].position = new\_position

bees[i].cost = new\_cost

# Selección de la mejor posición global

best\_bee = min(bees, key=lambda x: x.cost)

# Imprimir el costo y la posición de la mejor abeja en la época actual

print(f"Epoch {epoch+1}: Best Cost = {best\_bee.cost}, Best Position = {best\_bee.position}")

print(f'\nEVALUACION DE LAS ABEJAS EXPLORADORES EN LA EPOCA {epoch}')

# Etapa de exploración de abejas observadoras

for i in range(num\_bees):

# Elegir una abeja aleatoria distinta de la actual

other\_bee\_index = random.choice([idx for idx in range(num\_bees) if idx != i])

other\_bee = bees[other\_bee\_index]

# Generar una posición candidata cercana a la posición de la abeja seleccionada

new\_position = other\_bee.position + random.uniform(-8, 10) \* (other\_bee.position - bees[i].position)

# Asegurar que la nueva posición esté dentro de los límites

new\_position = max(min\_val, min(new\_position, max\_val))

# Calcular el costo de la nueva posición

new\_cost = objective\_function(new\_position)

# Actualizar la posición si la nueva posición es mejor

if new\_cost < bees[i].cost:

bees[i].position = new\_position

bees[i].cost = new\_cost

print(f'ESTE FUE LA MEJOR POSICIÓN {bees[i].position} DE LA ABEJA EXPLORADORA {i} EN LA EPOCA {epoch}')

# Encontrar la mejor solución global

best\_solution = min(bees, key=lambda x: x.cost)

# Imprimir la solución óptima

print("\nOptimal Solution:")

print(f"Cost = {best\_solution.cost}")

print(f"Position = {best\_solution.position}")

# Parámetros del algoritmo

min\_val = -15

max\_val = 35

num\_bees = 15

max\_epochs = 500

# Ejecutar el algoritmo de abejas

bee\_algorithm(min\_val, max\_val, num\_bees, max\_epochs)

Repositorio en GitHub: [Ver repositorio](https://github.com/JunniorRavelo/Artificial-Bee-Colony-ABC-.git)