

Trabajo Fin de Grado

DESARROLLO DE UNA HERRAMIENTA PARA LA APLICACIÓN DE ALGORITMOS DE ENJAMBRES COMO TÉCNICA DE CUANTIFICACIÓN DE COLOR



Autor:

Ángel Casado Nogueras

Tutores:

María Luisa Pérez Delgado Jesús Ángel Román Gallego

Junio 2025

Índice

[1. Motivaciones 5](#_Toc199358850)

[2. Introducción 6](#_Toc199358851)

[2.1 Representación de imágenes en color 6](#_Toc199358852)

[2.2 Los modelos de color 7](#_Toc199358853)

[3. El problema de la cuantificación de color 7](#_Toc199358854)

[3.1 Definición del problema 7](#_Toc199358855)

[3.2 Aplicaciones prácticas 9](#_Toc199358856)

[4. Los algoritmos de enjambres. 10](#_Toc199358857)

[4.1 Características generales. 10](#_Toc199358858)

[4.2 Algoritmo de optimización por enjambre de partículas (PSO) 11](#_Toc199358859)

[4.2.1 Introducción 11](#_Toc199358860)

[4.2.3 Variables 11](#_Toc199358861)

[4.2.4 Definición del algoritmo 12](#_Toc199358862)

[4.2.5 Parámetros PSO 14](#_Toc199358863)

[4.3 Luciérnagas (Firefly) 15](#_Toc199358864)

[4.3.1 Introducción 15](#_Toc199358865)

[4.3.3 Variables 15](#_Toc199358866)

[4.3.4 Definición del algoritmo 15](#_Toc199358867)

[4.4 Lobos / Grey Wolf optimization (GWO) 17](#_Toc199358868)

[4.4.1 Introducción 17](#_Toc199358869)

[4.4.2 Fundamentos 17](#_Toc199358870)

[4.4.3 Definición del algoritmo 19](#_Toc199358871)

[4.4.4 Variables 21](#_Toc199358872)

[4.5 El algoritmo de Ballenas (Whale Swarm Algorithm - WSA) 22](#_Toc199358873)

[4.5.1 Introducción 22](#_Toc199358874)

[4.5.2 Fundamentos 22](#_Toc199358875)

[4.5.3 Definición del algoritmo 23](#_Toc199358876)

[4.6 Abejas – Algoritmo Colonia de Abejas Artificiales (ABC) 25](#_Toc199358877)

[4.6.1 Introducción 25](#_Toc199358878)

[4.6.3- Variables 25](#_Toc199358879)

[4.6.4- Fundamentos del algoritmo 26](#_Toc199358880)

[5. Tecnologías empleadas 30](#_Toc199358881)

[5.1 Python 30](#_Toc199358882)

[5.2 Numphy 30](#_Toc199358883)

[5.3 OpenCv 31](#_Toc199358884)

[5.4 Scikit-learn 31](#_Toc199358885)

[5.5 Scikit-image 31](#_Toc199358886)

[5.6 Entornos virtuales 31](#_Toc199358887)

[5.7 Control de versiones 32](#_Toc199358888)

[5.8 Visual Studio Code 32](#_Toc199358889)

[5.9 Otros 32](#_Toc199358890)

[6. Pruebas 33](#_Toc199358891)

[6.1 MSE 33](#_Toc199358892)

[6.2 MAE 34](#_Toc199358893)

[6.3 SSIM 34](#_Toc199358894)

[6.4 UQI 34](#_Toc199358895)

[6.5 Tablas de resultados 34](#_Toc199358896)

[7. Desarrollo del programa 35](#_Toc199358897)

[5.1 Archivo main o ejecutor 35](#_Toc199358898)

[5.2 Algoritmos 39](#_Toc199358899)

[5.2.1 PSO 39](#_Toc199358900)

[5.2.2 Luciérnagas 43](#_Toc199358901)

[5.2.3 Lobos 48](#_Toc199358902)

[5.2.4 Abejas 52](#_Toc199358903)

[5.3 Funciones 61](#_Toc199358904)

[5.3.1 pintaImagen 61](#_Toc199358905)

[5.3.2 preparaImagen 62](#_Toc199358906)

[5.3.3 generaCuantizada 63](#_Toc199358907)

[5.3.4 getMse 64](#_Toc199358908)

[5.3.5 getMae 64](#_Toc199358909)

[5.3.6 getSsim 64](#_Toc199358910)

[5.3.7 getMssim 65](#_Toc199358911)

[8. Trabajo futuro 66](#_Toc199358912)

[9. Conclusiones 67](#_Toc199358913)

[10.Bibliografía 67](#_Toc199358914)

Índice de figuras

Ilustración 1 Imagen resultante con 32 colores 7

Ilustración 2 Imagen original [19] 7

Ilustración 3 Pirámide 16

Ilustración 4 Lobos Grises[20] 20

Ilustración 5 Logo Python [21] 29

Ilustración 6 Logo Numpy[22] 29

Ilustración 7 Logo GitHub[23] 31

Ilustración 8 JSON de argumentos 31

Ilustración 9 Logo VisualStudioCode [24] 31

Ilustración 10 Argumentos 34

Ilustración 11 Obtener ruta de los directorios de imágenes 35

Ilustración 12 Comprobaciones de rutas e imágenes. 35

Ilustración 13 Comprobaciones extra 36

Ilustración 14Diccionarios Python de algoritmos y funciones 36

Ilustración 15 Método ejecutor de la herramienta 37

Ilustración 16 Constantes y directorios 37

Ilustración 17 Bucles 38

Ilustración 18 Sistema de checkpoints 38

Ilustración 19 Bucle Imagenes 39

Ilustración 20 Comienzo de la prueba 39

Ilustración 21 Ejecución de la herramienta 39

Ilustración 22 Fin del script 40

Índice de tablas

[Tabla 1 Tiempos PSO 34](#_Toc200903711)

[Tabla 2 Tiempos FA 34](#_Toc200903712)

[Tabla 3 Tiempos ABA 34](#_Toc200903713)

[Tabla 4 Tiempos GWO 35](#_Toc200903714)

[Tabla 5 Tiempos WOA 35](#_Toc200903715)

# 1. Motivaciones

El desarrollo de este trabajo de fin de grado nace del creciente interés en las técnicas de optimización inspiradas por la naturaleza, en particular los algoritmos de enjambre, que han demostrado ser métodos bastante eficaces y versátiles en la solución de problemas complejos en diversas áreas.

También ha sido motivado por la asignatura de Sistemas Inteligentes con la cual personalmente disfruté mucho cursándola. En ella pudimos probar otro lenguaje de programación como Python, comprobar las diferencias que existen entre C y Python, e implementar código con un objetivo algo más visual que en el resto de la carrera.

# 2. Introducción

La cuantificación de color en imágenes es un problema complejo en estos tiempos en los que nos comunicamos frecuentemente con el uso de imágenes.

El objetivo principal es reducir el número de colores de una imagen a un conjunto más pequeño de colores representativos, a los que se llama **paleta cuantizada** de colores, con la menor pérdida de calidad posible. La imagen resultante, que se denomina **imagen cuantizada**, debe ser reconocible y lo más parecida posible visualmente a la original, pero con una representación mucho más simple y eficiente.

Puede parecer un proceso simple a primera vista, pero representa varios desafíos y requiere del uso de técnicas avanzadas, como algoritmos de enjambres, para obtener buenos resultados usando un tiempo de cómputo razonable. Algunas de las razones que explican por qué la cuantificación de color en imágenes es un problema complejo son:

* **Gran volumen de datos:** Ya que las imágenes que se procesan pueden contener miles de pixeles y cada píxel está representado por varios datos, procesar todos estos datos de manera eficiente y efectiva es un gran desafío.
* **Preservación de detalles importantes:** para la cuantificación de color es esencial preservar los detalles importantes de la imagen, como bordes y características distintivas. La simplificación excesiva puede llevar a la pérdida de información crítica y degradar la calidad visual de la imagen.
* **Selección de colores representativos:** Elegir los colores que representarán la imagen cuantizada de manera precisa es una tarea compleja. La selección de estos colores debe hacerse de manera inteligente para garantizar que la imagen cuantizada sea lo más fiel posible a la original.
* **Mantenimiento de la apariencia visual:** La reducción de color debe lograr un equilibrio entre la reducción del número de colores propiamente dicha y la retención de la apariencia visual general de la imagen. Los cambios agresivos en los colores pueden hacer que la imagen resultante sea irreconocible o menos atractiva.
* **Optimización de la eficiencia:** Procesar imágenes en tiempo real o en aplicaciones donde la velocidad es crucial (como la transmisión de video) requiere algoritmos de reducción de color eficientes que puedan realizar cálculos rápidos y precisos.
* **Evaluación de la calidad:** Medir la calidad de la imagen resultante es subjetivo y puede depender de la percepción humana. Evaluar y comparar algoritmos en términos de calidad de reducción de color es un aspecto importante de la investigación en este campo.

## 2.1 Representación de imágenes en color

La representación de imágenes en color en un ordenador es un proceso fundamental que implica codificar y almacenar la información de color de una imagen para luego procesarla, visualizarla y manipularla. Para lograr esto, los ordenadores utilizan modelos de color, siendo el modelo RGB (Red, Green, Blue) uno de los más comunes.

En el modelo RGB, cada píxel de una imagen se representa como una combinación de tres canales de color: rojo, verde y azul. Cada canal puede tomar valores entre 0 y 255 que representan la intensidad de ese color en un píxel específico.

La combinación de estos tres canales de color en cada píxel permite crear una gran cantidad de colores y tonalidades (16.777.216 colores posibles), lo que nos permite representar imágenes en color de alta calidad. La información de color de cada píxel se almacena en una matriz tridimensional, donde las dos primeras dimensiones representan la posición espacial del píxel en la imagen y la tercera dimensión almacena los valores de intensidad de rojo, verde y azul.

Esta representación de color en RGB es la base para todas las operaciones de procesamiento de imágenes en color, incluida la reducción de color. Durante este proceso, el objetivo es reducir el número de colores de la imagen, manteniendo al mismo tiempo la apariencia visual deseada. Los algoritmos de enjambres nos ayudan a “escoger” los colores adecuados para preservar la calidad en la imagen reducida.

## 2.2 Los modelos de color

Ya que se ha hablado de cómo un ordenador representa las imágenes en color utilizando el modelo de color RGB, me parece importante hablar brevemente de otros modelos que existen.

Uno de los más simples es la **escala de grises** que tiene un solo canal, en el que representa valores de grises del blanco al negro. Pueden tener una profundidad de 8 o de 16 bits por píxel.

Otro de ellos es el **indexado** en el que las imágenes tienen una gama de colores reducida para que sus archivos sean más pequeños. Las imágenes indexadas por lo general se suelen usar en páginas web o en gráficos con pocos colores. Solo cuentan con un canal de 8 bits.

Otros dos modelos bastantes extendidos hoy en día son:

1. **CMYK:** Es un modelo de color de cuatro canales. Los valores representan el color cian, el magenta, el amarillo y el negro. Al igual que el RGB puede tener 8 o 16 bits por canal. Este modelo compone los colores de manera sustractiva, es decir, parte del blanco y va restando colores usando sus complementarios.

Este modelo es el que usan habitualmente las impresoras. A la hora de retocar las fotos en el ordenador nos servirá de gran ayuda para ver si los colores de la imagen que hemos trabajado en RGB serán los mismos que los impresos en CMYK ya que a veces no coinciden. Esto ocurre porque hay algunos colores RGB que no pueden conseguirse en CMYK.

1. **LAB:** En este modelo existen 3 canales, L, A y B. L representa la información tonal o luminancia (intensidad de la luz en la imagen) y A y B la información de color o crominancia (cómo los colores se combinan y varían en una imagen), siendo A la que añade los colores en la línea del rojo y el verde y B la que añade el contenido en amarillo o azul.[1]

# 3. El problema de la cuantificación de color

## 3.1 Definición del problema

Una imagen digital se forma con una cantidad de píxeles representados usando un modelo de color concreto. Cuando este modelo de color es el RGB, como ya se ha indicado, cada píxel se representa con 3 números enteros entre el 0 y el 255 por lo cual este espacio de color nos permite usar 2563 colores diferentes.

Como se ha dicho anteriormente la cuantificación de color trata de reducir este número de colores evitando la pérdida de información visual. Para conseguirlo realiza dos operaciones:

1. **Selección de colores representativos:** Esta operación implica la elección de un conjunto limitado de colores representativos que serán utilizados para reemplazar los colores originales de la imagen. Para seleccionar estos colores, se aplican algoritmos de agrupación de colores, uno de ellos es el algoritmo K-means, que agrupa los colores similares en clusters y utiliza los centroides de estos clusters como colores representativos. Este conjunto de colores representativos lo llamamos **paleta cuantizada**. Esta operación es esencial para simplificar la representación de color de la imagen y reducir el número de colores utilizados.
2. **Asignación de colores representativos a los píxeles:** Una vez que se han seleccionado los colores representativos, se lleva a cabo la operación de asignación de colores. En esta etapa, cada píxel en la imagen original se asigna al color representativo más cercano en términos de distancia de color seleccionado en la anterior operación. Esto implica calcular la diferencia de color entre el color original del píxel y los colores representativos de la paleta cuantizada y seleccionar el color representativo que minimice esta diferencia. Sin embargo, existen otros métodos de asignación de colores, y este es solo uno de ellos. Dependiendo del algoritmo o el objetivo, se podrían emplear técnicas alternativas para asignar los colores representativos a los píxeles. Esta operación es fundamental para reemplazar los colores originales por sus equivalentes representativos y, de esta manera, lograr la reducción de color en la imagen.

A continuación, en la Ilustración 1 se muestra una imagen del paquete CQ100 [18] y en la Ilustración 2 se muestra la imagen cuantizada resultante con 32 colores.

Ilustración 1 Imagen resultante con 32 colores

Ilustración 2 Imagen original [18]



## 3.2 Aplicaciones prácticas

La cuantificación de color tiene diversas aplicaciones prácticas en varios campos, tales como:

* **Procesamiento de imágenes y fotografía:** Se utiliza para reducir el número de colores en una imagen digital, lo que es útil para la compresión de imágenes y la optimización para dispositivos y medios con capacidad de color limitada.
* **Diseño gráfico y web:** Ayuda a los diseñadores a crear paletas de colores coherentes y limitadas para mantener la consistencia y mejorar la estética en diseños y sitios web.
* **Reconocimiento de patrones e identificación de objetos:** la cuantificación de color puede ayudar a simplificar la identificación y clasificación de objetos al reducir la complejidad de las imágenes, facilitando el reconocimiento de patrones.
* **Impresión y fabricación de textiles:** En la industria de la impresión y en la fabricación de textiles, la cuantificación de color es crucial para asegurar que los colores utilizados en los diseños sean reproducibles y consistentes en diferentes lotes de producción.
* **Cartografía y análisis geoespacial:** Se utiliza para simplificar la representación visual de mapas y datos geoespaciales, ayudando a resaltar características importantes y mejorar la legibilidad.
* **Análisis médico y de imágenes biomédicas:** En el campo de la medicina, la cuantificación de color se aplica en el análisis de imágenes biomédicas para mejorar la visualización y el diagnóstico de ciertas condiciones al resaltar variaciones sutiles en los tejidos.
* **Análisis de calidad y control en la industria alimentaria:** Se usa para evaluar la calidad y la madurez de alimentos basándose en su color, lo cual es especialmente útil en líneas de producción automatizadas.
* **Arte y restauración:** En el arte digital y la restauración de obras, la cuantificación de color puede ayudar a analizar y replicar los colores utilizados por los artistas originales, así como a detectar alteraciones o daños.
* **Seguridad y vigilancia:** La cuantificación de color puede mejorar el rendimiento de los sistemas de vigilancia al reducir la complejidad de las imágenes, facilitando la detección de movimientos o actividades sospechosas.
* **Investigación científica y análisis de datos:** En diversas áreas científicas, la cuantificación de color puede ser útil para visualizar y analizar datos, especialmente en representaciones gráficas de información compleja.

# 4. Los algoritmos de enjambres.

## 4.1 Características generales.

Los algoritmos de enjambres o inteligencia de enjambres es la disciplina que se ocupa de los sistemas naturales y artificiales compuestos por muchos individuos que se coordinan mediante el control descentralizado y la autoorganización. Se trata de un subcampo de la inteligencia artificial que se centra en los comportamientos colectivos que resultan de las interacciones de los individuos entre sí y con su entorno o medio ambiente al igual que lo hacen los enjambres naturales.

Algunos de los sistemas estudiados por la inteligencia del enjambre son las colonias de hormigas y termitas, los bancos de peces, las bandadas de pájaros o manadas de animales terrestres.

Ciertos artefactos humanos también caen en el dominio de la inteligencia de enjambre. En particular, algunos sistemas de múltiples robots, y también algunos programas de computadora que están escritos para abordar problemas de optimización y análisis de datos.[2]

Algunas propiedades de estos algoritmos son:

1. **La autonomía**: Cada agente (o partícula) en el enjambre opera de manera individual, tomando decisiones basadas en su información local y, posiblemente, en alguna información global disponible para el enjambre.
2. **Descentralización**: No existe un control centralizado que dirija las acciones de los agentes; en su lugar, el comportamiento global emerge de las interacciones locales entre los agentes y entre los agentes y su entorno.
3. **Distribución**: Los algoritmos de enjambres suelen ser inherentemente distribuidos, lo que los hace robustos y escalables, ya que el fallo de una partícula tiene un impacto limitado en el enjambre en su totalidad.
4. **Autoorganización**: Las interacciones entre las partículas y la aplicación de reglas sencillas dan lugar a un comportamiento colectivo "inteligente" y autoorganizado. Este comportamiento es emergente, es decir, surge de las acciones de las partículas individuales.
5. **Interacciones simples**: Las reglas que rigen las interacciones entre las partículas son simples, pero pueden dar lugar a comportamientos colectivos complejos y a la solución de problemas complejos.
6. **Retroalimentación positiva y negativa**: Los mecanismos de retroalimentación, tanto positiva como negativa, son muy importantes para el desarrollo y la estabilización de los patrones de comportamiento del enjambre. La retroalimentación positiva promueve la formación de estructuras o caminos para hallar una mejor solución mientras que la retroalimentación negativa ayuda a prevenir la saturación de datos o una mala solución rápida.
7. **Exploración y explotación**: Los algoritmos de enjambres equilibran entre la exploración del espacio de búsqueda para descubrir nuevas soluciones y la explotación de las mejores soluciones encontradas para afinarlas.
8. **Adaptabilidad**: La inteligencia de enjambre es altamente adaptable a cambios en el entorno o en los parámetros del problema, lo que permite al enjambre encontrar nuevas soluciones cuando las condiciones cambian.
9. **Robustez**: Los enjambres son robustos frente a fallos y variaciones, ya que la pérdida de algunas partículas generalmente no impide que el enjambre en su conjunto continúe funcionando eficazmente.

Algo que comparten todos los algoritmos de optimización es que realizan su búsqueda en un determinado espacio de búsqueda, en el caso de la cuantificación de color se puede definir claramente al estar determinado por el número de colores que se desea obtener en la paleta cuantizada, y cada color está representado por un vector de tres dimensiones que corresponden a las componentes RGB.

Por lo tanto, si se desea reducir la imagen a una paleta de colores, el espacio de búsqueda sería: dimensiones.

## 4.2 Algoritmo de optimización por enjambre de partículas (PSO)

### 4.2.1 Introducción

La **Optimización por Enjambres de Partículas** (conocida como **PSO**, por sus siglas en inglés, **Particle Swarm Optimization**) es una técnica de optimización/búsqueda.

Este algoritmo fue descrito alrededor de 1995 por Kennedy y Eberhart, y se inspira en el comportamiento de los enjambres de insectos en la naturaleza.

En este algoritmo se trabaja con una función objetivo que queremos optimizar, denotada comúnmente como f(x1…, xn), esta función devuelve una medida de lo buena que es una solución en cierto punto del espacio de búsqueda.

El objetivo de este algoritmo es encontrar los valores óptimos de las variables x1, x2, …, xn para maximizar (o minimizar) el valor de la función f. Esta función se denomina **función fitness.**[3]

### 4.2.3 Variables

* Se considera un conjunto de **I** individuos llamados en este algoritmo **partículas**. Cada una de las **I** partículas tendrá asociado un estado que se irá modificando con el tiempo. Estas partículas tendrán:
  + Posición (xi): representa el estado de la partícula, es una posible solución al problema.
  + Velocidad (vi): determina la actualización de la posición.
  + Mejor posición personal (bi): esta variable representa la mejor solución encontrada por la partícula durante la ejecución.
* La solución al problema viene dada por la mejor posición encontrada por el enjambre la cual llamamos la mejor posición global, representada como: g.

En este algoritmo las partículas (individuos) se mueven por el espacio de solución del problema guiadas por ellas mismas y por el conjunto de todo el enjambre.

Este movimiento hace variar su posición, velocidad y mejor posición personal.

### 4.2.4 Definición del algoritmo

**PSEUDOCODIGO**

Inicializar la población de individuos

**REPETIR**

Evaluar el fitness de cada individuo.

Actualizar la mejor solución personal de cada individuo.

Actualizar la mejor solución global.

Actualizar la velocidad y posición de cada individuo. ([Ec.1](#ec1pso), [Ec.2](#ec2pso))

**HASTA** (condición de parada)

**Inicializar la población de individuos**

Al principio del algoritmo se inicializan las variables de cada individuo su **posición** y su **velocidad.**

Su posición será un valor aleatorio dentro del espacio de búsqueda y su velocidad será un valor aleatorio entre [vmin , vmax ], que son dos parámetros del algoritmo que se definen a priori.

**Evaluar el fitness de cada individuo**

Para calcular este valor se aplica la función objetivo del problema a la posición de la partícula. En el caso de la cuantificación de color se utilizarán funciones tales como MSE (error cuadrático medio), MAE (Error absoluto medio), SSIM (Índice de similitud estructural) o MS-SSIM (Índice de similitud estructural multiescala) (a partir de ahora función objetivo).

Este valor llamado fitness determina la calidad de la solución cuya posición representa.

**Actualizar la mejor solución personal de cada individuo.**

Si al calcular el fitness un individuo encuentra una mejor solución personal (es decir, encuentra una mejor posición lo que resulta en un mejor fitness) que la que tuviese almacenada para ella hasta el momento, actualiza la mejor solución con el valor de ésta.

Se da un nuevo valor a bi

**Actualizar la mejor solución global.**

Con la solución global ocurre lo mismo que con la mejor solución personal de cada individuo: si se ha encontrado una solución que mejora a la que hubiese almacenada como mejor solución global, se guarda como nueva mejor solución global.

Se da un nuevo valor a g.

**Actualizar la velocidad y posición de cada individuo.**

En este paso se realizan dos procedimientos:

1. *Actualizar velocidad de cada individuo.*

Se calcula una nueva velocidad de cada individuo i:

Ec.1

Cuyos parámetros son:

**e1, e2:** valores aleatorios en [0, 1].

**w:** inercia.

**f1:** parámetro cognitivo.

**f2:** parámetro social.

Esta actualización incluye tres componentes que son:

-Velocidad anterior:

Donde w (inercia) debe seleccionarse atendiendo a que si es grande se realizará una exploración global del espacio de solución, si es pequeña será exploración local. Se aconseja que este parámetro decrezca con las iteraciones.

-La distancia a la mejor posición personal:

En esta componente encontramos dos parámetros:

**e1:** será un valor aleatorio entre [0, 1].

**f1:** el parámetro cognitivo que determina la importancia que se da a la experiencia propia de cada individuo.

-La distancia a la mejor solución global:

En esta componente tenemos los parámetros:

**e2:** será un valor aleatorio entre [0, 1].

**f2:** el parámetro social que determina la importancia que se da a la experiencia del enjambre.

Con el valor calculado de la velocidad, se debe ajustar al intervalo válido de la velocidad, es decir, nuestro nuevo valor: deberá estar entre .

Para ello:

-

-

1. *Actualizar posición de cada individuo.*

Se calcula la nueva posición de cada individuo i.

Ec.2

Este nuevo valor para la posición de cada individuo depende de la actual posición y de la nueva velocidad que se acaba de calcular.

### 4.2.5 Parámetros PSO

Como se ha hablado antes en el PSO existen varios parámetros principales que influyen en el comportamiento y en la eficacia del algoritmo, son: el coeficiente de **inercia**, y los parámetros **cognitivo** y **social**.

**Inercia:**

Este coeficiente afecta a la velocidad a la que los individuos se mueven en el espacio de búsqueda. Ayuda a controlar los resultados de la velocidad anterior del individuo sobre su velocidad actual. Un valor alto ayuda a que el individuo mantenga su dirección y velocidad anterior, lo que se traduce en una mejor exploración global. Por otro lado, un valor bajo hace que el individuo pierda velocidad por lo que ayuda a una exploración más local y detallada.

**Parámetro cognitivo:**

También conocido como parámetro personal o local, determina el grado en el que cada individuo se ve influido por sí mismo. Influye en cuánto se moverá un individuo hacia la mejor posición encontrada individualmente. Un valor alto de este componente dará a cada individuo una mayor motivación para seguir su propio camio hacia lo que percibe como una mejor solución, dando así lugar a una mayor variedad de soluciones exploradas por el enjambre.

**Parámetro social:**

Determina el grado de influencia de la mejor posición encontrada por todo el enjambre sobre el movimiento de cada individuo. Valores más altos de este componente resultan en una mayor convergencia del enjambre hacia las áreas del espacio de búsqueda que parecen más prometedoras según la experiencia del enjambre, es decir, los individuos preferirán moverse hacia la mejor solución encontrada por el conjunto de individuos.

## 4.3 Luciérnagas (Firefly)

### 4.3.1 Introducción

El algoritmo firefly, es un algoritmo metaheurístico de optimización, inspirado en el comportamiento del centelleo de las luciérnagas [4]. El propósito primario de una luciérnaga es generar destellos de luz para actuar como sistema de señal para atraer a otras luciérnagas. Xin-She Yang formuló este algoritmo con las siguientes premisas:

1. Todas las luciérnagas son "unisexuales", de modo que cualquier luciérnaga individual será atraída por todas las demás.
2. La atracción es proporcional a su brillo, y para cualquier par de luciérnagas, la menos brillante será atraída por (y por lo tanto se desplazará hacia) la más brillante; aun así, la intensidad (el brillo aparente) decrece cuando aumenta la distancia entre ambas;
3. Si no hay luciérnagas más brillantes que una dada, esta se mueve aleatoriamente.

El brillo es asociado con los valores de una función objetivo.

### 4.3.3 Variables

En este algoritmo se considera un conjunto de I individuos (luciérnagas). Cada uno de estos individuos representa una posible solución al problema. La calidad de esa solución dependerá del **brillo** que esté asociado a cada individuo.

Posición →

Brillo →

En este contexto el **brillo** de cada individuo será el fitness de este algoritmo, es decir, el individuo que posea el mayor brillo será la mejor solución al problema. Como en este algoritmo los individuos se mueven hacia el individuo con un mayor brillo, todos ellos se acercan a la mejor solución, menos el individuo más brillante.

La **atracción** entre los individuos disminuye con la distancia debido a la absorción de la luz. Este valor normalmente va cambiando para representar la disminución (visibilidad) del brillo con la distancia.

### 4.3.4 Definición del algoritmo

**PSEUDOCODIGO**

Generar la población inicial de luciérnagas

**REPETIR**

Mover cada luciérnaga hacia las más brillantes ([Ec.3](#ec1Luciernagas) [Ec.4](#ec2Luciernagas))

Mover la luciérnaga más brillante [Ec.5](#ec3Luciernagas)

Actualizar el brillo de las luciérnagas

Ordenarlas por brillo y encontrar la mejor

**HASTA** (condición de parada)

**Generar la población inicial de luciérnagas**

En este primer paso repartimos los individuos aleatoriamente en el espacio de búsqueda y calculamos el brillo inicial de cada uno.

Por cada iteración:

**Mover cada luciérnaga hacia las más brillantes**

Cada luciérnaga (excepto la más brillante) mueve su posición hacia la más brillante, debido a esto se calcula una nueva posición para cada luciérnaga, en función de su posición previa y de la atracción ejercida por las otras luciérnagas.

Un individuo **I** se mueve hacia otro individuo **k**, que es más brillante que él, siguiendo esta ecuación:

Ec.3

Donde:

-> es la siguiente posición del individuo i

-> es la posición actual del individuo i

-> es el atractivo del individuo k sobre el i

-> un valor aleatorio

Donde β(rik) se calcula mediante:

Ec.4

-> atractivo a distancia cero

->coeficiente de absorción de la luz del medio

rij -> distancia entre los individuos i y j

**Mover la luciérnaga más brillante**

El individuo más brillante no se ve atraído por ningún otro, por lo que se mueve aleatoriamente.

Su nueva posición será:

Ec.5

**Actualizar el brillo de las luciérnagas**

En este paso se calcula el fitness (brillo) de cada individuo mediante la función objetivo del problema.

**Ordenarlas por brillo y encontrar la mejor**

En este último paso se ordenan los individuos por brillo y se determina el mejor (el que más brillo tenga). Para ello se utilizan diversos algoritmos de ordenación comunes (esta elección dependerá del número de individuos, la complejidad del algoritmo, y si el conjunto de datos esta parcialmente ordenado).

## 4.4 Lobos / Grey Wolf optimization (GWO)

### 4.4.1 Introducción

Este algoritmo de optimización metaheurística está basado en el comportamiento social y de caza de los lobos grises. Fue desarrollado por **Seyedeli Mirjalili** en 2014, simula tanto la estructura social de estos lobos como sus tácticas de caza para resolver complejos problemas de optimización. Estos lobos son conocidos por su comportamiento de caza estratégico y su jerarquía social, estos componentes se incluyen en este algoritmo para resolver estos problemas.

En esta jerarquía social, los individuos se clasifican en varias categorías: Alfas, Betas, Deltas y Omegas. Los Alfas son los que lideran la manada, estos toman las decisiones sobre la caza, el movimiento del grupo y su descanso. Los Betas, que son los colideres, ayudan a los Alfas en la toma de decisiones y en la comunicación con el resto de los individuos de la manada. Los Deltas ocupan un rango inferior, estos se ocupan de cuidar o vigilar. Por último, los Omegas están en la base de la pirámide jerárquica, estos siguen las decisiones de los miembros de rango superior sin cuestionar sus decisiones.

Este algoritmo utiliza esta jerarquía para modelar la cooperación entre los lobos en la resolución de problemas de optimización. Los mejores candidatos que son los Alfas lideran la búsqueda, estos son seguidos por los Betas y los Deltas, mientras que los Omegas siguen las direcciones establecidas por los miembros superiores. Durante el proceso de optimización, todos los lobos, que son todas las posibles soluciones, van ajustando sus posiciones respecto a los lobos de mejor desempeño en su entorno (espacio de búsqueda).

Este enfoque le permite al GWO explorar eficazmente el espacio de búsqueda y explotar las áreas más prometedoras que se acerquen a las mejores soluciones encontradas. Posee un gran equilibrio entre la exploración y la explotación de las soluciones, y una gran capacidad para no quedarse estancado en mínimos locales, el GWO ha demostrado ser efectivo en una gran variedad de aplicaciones prácticas, desde la ingeniería hasta la ciencia de datos. Desde sus inicios, este algoritmo ha generado un gran interés en la comunidad de investigación por su robustez y rendimiento superior en comparación con otras técnicas de optimización.[5]

### 4.4.2 Fundamentos

El GWO introduce I lobos en la manda, desde aquí llamados I individuos.

-Cada individuo representa una posible solución al problema.

-La calidad o fitness de esa solución se calcula mediante la función objetivo del problema.

Esta jerarquía se representa de la siguiente manera:

Los tres mejores lobos de la manada estarán en los tres niveles superiores de la jerarquía, es decir, en cada uno de estos tres niveles habrá un solo lobo.

-El mejor lobo (mejor solución) se denota como α. ->

-El segundo mejor lobo se denota como β. ->

-El tercer mejor lobo se denota como δ. ->

-El resto de los lobos se consideran lobos ω. ->

Ilustración 3 Pirámide

El proceso de optimización está representado por la operación de caza, en esta actividad los lobos de los tres primeros niveles guían, mientras que el resto siguen a éstos.

Este algoritmo posee varios coeficientes sociales (a, A y C).

El parámetro a o **factor de exploración** es el que guía el proceso de búsqueda. Este parámetro varía linealmente desde 2 hacia 0 con las iteraciones y se utiliza para calcular dos vectores aleatorios (A y C) que serán los condicionantes de la exploración/explotación del algoritmo.

Estos vectores al depender de a también disminuirán con las iteraciones. Estos se calculan de la siguiente manera:

Ec.6 Ec.7

Donde:

: son vectores aleatorios en [0, 1].

: es el parámetro del algoritmo (factor de exploración)

Cuando el valor absoluto de estos vectores es mayor que uno se entiende que los individuos divergen respecto de la presa, se dice que están en fase de exploración.

Al converger a la presa, este valor será menor que uno, en este caso están en fase de explotación.[6]

### 4.4.3 Definición del algoritmo

En el GWO la búsqueda de la presa, es decir, la solución del problema se inicia con la creación de una población de I individuos al azar, igual que en el resto de los algoritmos.

Después de esto se realiza un proceso iterativo en el cual:

1. Los lobos: alfa, beta y delta, estiman la posición probable de la presa.
2. Cada lobo del grupo actualiza su distancia respecto de la presa.
3. Se decrementa el parámetro a.

**PSEUDOCODIGO**

Inicializar la población de lobos

Calcular el fitness de los lobos

Determinar los lobos alfa, beta y delta

**REPETIR**

Actualizar la posición de cada lobo [Ec.8](#ec3Lobos)

Ajustar el parámetro a [Ec.9](#ec10Lobos)

Calcular el fitness de los lobos

Determinar los lobos alfa, beta y delta

**HASTA** (condición de parada)

En el primer paso, **inicializar la población de individuos**, se escogen unos valores aleatorios del espacio de solución para la posición de cada individuo Xi

Aparte de esto también se inicializa el parámetro a, este factor de exploración comienza con el valor de 2.

En el segundo paso, se calcula el **fitness** de cada individuo, esto se hace en base al valor de la función objetivo en Xi

Lo siguiente es decidir los individuos alfa, beta y delta que serán los individuos con mejor fitness.

Xα, Xβ y Xδ

En el bucle, el primer paso será actualizar la posición de cada individuo. Esta nueva posición será:

Ec.8

Ec.4 Ec.7

Ec.5 Ec.8

Ec.6 Ec.9

En donde:

T: es la iteración actual.

Ai, Ci (i=1, 2, 3): vectores de coeficientes, calculados usando la [Ec6](#ec1Lobos) y la [Ec7](#ec2Lobos).

En el siguiente paso, se **actualiza el parámetro** del algoritmo.

El factor de exploración (a) se reduce linealmente en el intervalo [2,0] al avanzar las iteraciones, según:

Ec.9

En donde:

: es la iteración actual del algoritmo.

: es el número máximo de iteraciones del algoritmo.[7]

Para **calcular el fitness de cada individuo** se aplica la función objetivo del problema.

Por último, para **encontrar los mejores individuos** (alfa, beta y delta) se toman los 3 individuos con mejor fitness.

### 4.4.4 Variables

Como en cualquiera de estos algoritmos tenemos **el tamaño de la población** de individuos (I), en este caso es el número de lobos que existen en la población. Un mayor número de individuos puede mejorar la exploración del espacio de búsqueda, pero a su vez aumenta el costo computacional.



Ilustración 4 Lobos Grises[20]

En el GWO existen 3 parámetros o coeficientes sociales los cuales ya se han mencionado que son a, A y C.

* a: Disminuye linealmente de 2 a 0 a lo largo de las generaciones y se utiliza para calcular los coeficientes A y C.
* A: Calculado de la forma ([Ec6](#ec1Lobos)), es un vector aleatorio cuyos valores están entre el [0,1]. Este coeficiente afecta a la amplitud del movimiento del individuo (lobo) hacia o desde la presa.
* C: Calculado de la forma ([Ec7](#ec2Lobos)), este coeficiente influye en el reconocimiento y la localización de la presa, proporcionando un elemento de aleatoriedad en el acercamiento de los lobos a la presa.

Jerarquía de Lobos (Alfa, Beta, Delta)

* Alfa (α): El lobo líder, que posee la mejor solución encontrada hasta el momento.
* Beta (β): El segundo mejor lobo, apoya al Alfa en la toma de decisiones y mantiene la disciplina dentro del grupo.
* Delta (δ): El tercero en la jerarquía, juega un rol subordinado pero esencial, facilitando la comunicación entre el Alfa y Beta con el resto de la manada.
* Omega (ω): El resto de los lobos, siguen las direcciones de los lobos con rango superior.

De la misma manera que el resto de los algoritmos, el GWO también intenta optimizar una función, llamada función objetivo que es la que determina la calidad de cada solución.

Por último, los criterios de terminación del problema, que son las condiciones bajo las cuales el algoritmo debe finalizar, ya pueden ser una solución concreta, el número máximo de iteraciones o una limitación del tiempo de ejecución.

## 4.5 El algoritmo de Ballenas (Whale Swarm Algorithm - WSA)

### 4.5.1 Introducción

El WSA parte de la idea del WOA (Whale Optimization Algorithm), este algoritmo inspirado en el comportamiento de caza de las ballenas jorobadas fue desarrollado por Seyedali Mirjalili en 2016.

El WOA simula la técnica de burbujeo empleada por estos animales, es una estrategia predadora que utiliza la creación de burbujas en círculos o redes alrededor de su presa antes de atraparla. Este comportamiento es interesante desde un punto de vista de optimización, ya que engloba tanto los elementos de exploración como de explotación, que son las claves para la eficacia en la búsqueda de soluciones optimas en espacios complejos y multidimensionales.

Este algoritmo posee una estructura simple pero potente que permite resolver una amplia gama de problemas de optimización en ingeniería, informática y más allá. La adaptabilidad y eficiencia del WOA se deben a su capacidad para imitar la dinámica natural de estas ballenas, ajustando sus posiciones en el espacio de búsqueda de manera que maximiza las posibilidades de localizar y converger hacia un óptimo global.

### 4.5.2 Fundamentos

El WOA se basa en el comportamiento de caza de las ballenas jorobadas, particularmente en la técnica conocida como "burbujeo", donde las ballenas crean burbujas en círculos concéntricos alrededor de su presa. Esta técnica no solo confina a la presa, sino que también reduce sus vías de escape, facilitando a la ballena capturarla eficientemente.

Modelo Matemático del WOA

El WOA utiliza dos mecanismos principales basados en este comportamiento: la encerrona por burbujeo y los movimientos en espiral hacia la presa. La elección entre estos dos comportamientos se modela mediante un enfoque probabilístico que depende de la proximidad de la solución candidata al mejor candidato actual (la presa).

Encerrona por Burbujeo:

En este enfoque, las posiciones de las ballenas se ajustan según la posición del mejor candidato actual. Donde la posición se mueve hacia el mejor candidato o se aleja ligeramente de él basándose en un coeficiente aleatorio.

Movimiento en Espiral:

Este comportamiento simula el movimiento en espiral de las ballenas alrededor de su presa. Se utiliza una ecuación helicoidal para actualizar la posición de la ballena en dirección a la presa.

Características del WOA

-Exploración y Explotación:

El WOA equilibra eficazmente la exploración (buscando en nuevas áreas del espacio de búsqueda) y la explotación (intensificando la búsqueda cerca del óptimo conocido). La alternancia entre el comportamiento de encerrona por burbujeo y el movimiento en espiral permite este equilibrio, haciendo que el WOA sea robusto en encontrar soluciones globales.

-Flexibilidad y Aplicabilidad:

El algoritmo se ha aplicado en una variedad de problemas, desde optimización de funciones continuas hasta problemas de diseño industrial, demostrando su versatilidad y efectividad.

-Simplicidad de Implementación:

A pesar de su eficacia, el WOA es relativamente simple de implementar comparado con otros algoritmos metaheurísticos, lo que lo hace accesible para investigadores y profesionales.[8]

### 4.5.3 Definición del algoritmo

En este algoritmo se comienza por inicializar una población de I individuos, seguido por la evaluación de estos individuos basada en la función objetivo y se actualizan sus posiciones.

**PSEUDOCODIGO**

Inicializar la población de individuos.

**REPETIR**

PARA i+1 HASTA número de individuos HACER

Se actualiza a, A, C y p. [Ec.10](#ec1woa) [Ec.11](#ec2woa) [Ec.12](#ec3woa)

SI

SI

Se actualiza la posición respecto a la mejor solución. [Ec.13](#ec4woa) [Ec.14](#ec14)

SI NO

Se selecciona un individuo aleatorio y se actualiza su posición. [Ec.15](#ec6woa) [Ec.16](#ec7woa)

FIN SI

SI NO

Se mueve el individuo en un camino en espiral alrededor de la mejor solución. [Ec.17](#ec8woa)

FIN SI

Se actualiza la mejor solución global.

FIN-PARA

**HASTA** (condición de parada)

**Inicialización de la población**

Para ello se inicializan todos los individuos con posiciones aleatorias en el espacio de búsqueda permitido, después se calcula el fitness de cada individuo según la función objetivo y por último se calcula la mejor solución (individuo con mejor fitness).

También en este paso se inicializan los parámetros del algoritmo, el parámetro comienza con un valor de 2 y el como 1.

Después de este primer paso se actualizan los parámetros de este algoritmo para cada individuo siguiendo las siguientes ecuaciones.

**Actualización de** :

Ec.10

Donde:

es la iteración actual.

es el número total de iteraciones permitidas.

**Actualización de A**

Ec.11

Donde:

es el parámetro anteriormente calculado

r es un vector aleatorio en [0,1]

**Actualización de C**

Ec.12

Donde:

r es un vector aleatorio en [0,1]

**Se actualiza la posición respecto a la mejor solución:**

Se calcula el vector D

Ec.13

Donde:

C es el parámetro calculado anteriormente mediante la Ec.12.

X\* es la posición del mejor individuo

X es la posición del individuo que se está evaluando

Con D se calcula la nueva posición del individuo

Ec.14

**Se selecciona un individuo aleatorio y se actualiza su posición**, para ello se aplica la ecuación:

Para la cual se aplican esta ecuación para calcular

Ec.15

Ec.16

Donde:

es un individuo seleccionado aleatoriamente.

**Se mueve el individuo en un camino en espiral alrededor de la mejor solución** mediante esta ecuación que simula el movimiento en espiral hacia la presa:

Ec.17

Dónde:

es un parámetro que usa este algoritmo para definir el movimiento en espiral

es un número aleatorio en el intervalo [-1,1]

En la Ec.17 se utiliza la siguiente ecuación para calcular

## 4.6 Abejas – Algoritmo Colonia de Abejas Artificiales (ABC)

### 4.6.1 Introducción

El Algoritmo de Colonia de Abejas Artificiales (Artificial Bee Colony, ABC) es una técnica de optimización metaheurística que fue desarrollada por Karaboga en 2005. Inspirado por el comportamiento forrajero de las abejas, el ABC simula las dinámicas de búsqueda de comida de una colmena de abejas para encontrar soluciones óptimas en problemas complejos. Este algoritmo se basa en el modelo de división del trabajo y el reclutamiento mutuo entre abejas, características que permiten una exploración eficiente del espacio de soluciones.

Este proceso iterativo de exploración y explotación sigue hasta alcanzar una condición de terminación. Este algoritmo ha demostrado ser efectivo en problemas de optimización numérica y ha sido aplicado en muchos campos como la ingeniería, la investigación operativa y la inteligencia artificial. La simplicidad de este algoritmo sumada a su potente capacidad de optimización ha ayudado en gran parte a su popularidad y gran aceptación en la comunidad científica.[9], [10]

### 4.6.3- Variables

En el algoritmo ABC se define un conjunto de **F fuentes de alimento**, donde cada fuente representa una posible **solución al problema de optimización**. La posición de cada fuente está asociada a una solución y se considera como una variable clave en la búsqueda de soluciones óptimas.

* **Posición de la fuente**: Representa una solución factible dentro del espacio de búsqueda.
* **Néctar o calidad de la fuente**: La calidad o **fitness** de una fuente es una medida de su valor en términos de la función objetivo, es decir, cuanto mayor es la calidad de la fuente, mejor es la solución que representa.
* **Agotamiento de la fuente**: A medida que se explota una fuente, su **valor de calidad puede no mejorar**. Esto se modela a través de una variable que cuenta los intentos fallidos para mejorar dicha fuente. Si esta variable supera un cierto umbral, la fuente se considera agotada y se reemplaza por una nueva fuente aleatoria.

En el ABC, existen tres tipos de abejas que interactúan con estas fuentes de alimento:

* **Abejas exploradoras**: Se encargan de buscar nuevas fuentes de manera aleatoria, explorando áreas no visitadas previamente.
* **Abejas empleadas**: Se asignan a fuentes específicas y exploran en sus alrededores en busca de mejores soluciones.
* **Abejas observadoras**: Evalúan la información proporcionada por las abejas empleadas y seleccionan las fuentes más prometedoras para explotarlas.

Esta interacción entre los diferentes tipos de abejas y las fuentes de alimento permite al algoritmo buscar, explotar y reemplazar soluciones de manera eficiente a lo largo de las iteraciones.

### 4.6.4- Fundamentos del algoritmo

El algoritmo ABC sigue una estructura cíclica, donde diferentes tipos de abejas realizan operaciones sobre las fuentes de alimento, buscando mejorar la solución óptima. A continuación, se presenta el pseudocódigo simplificado que resume las principales etapas del algoritmo:

Inicializar el conjunto de fuentes de alimento

**REPETIR**

Operaciones de abejas empleadas [Ec.19](#ec2Abejas)

Operaciones de abejas observadoras [Ec.20](#ec3Abejas)

Operaciones de abejas exploradoras

Actualizar la mejor solución hasta el momento

**HASTA QUE** (condición de parada)

**Inicialización de las fuentes de alimento**

El proceso de inicialización consiste en seleccionar F fuentes de alimento dentro del espacio de búsqueda. A cada fuente se le asigna una posición aleatoria en dicho espacio, de acuerdo con la siguiente fórmula:

Ec.18

Dónde:

: número aleatorio en el intervalo [0, 1]

y : límites inferior y superior de la dimensión j del espacio de búsqueda.

: posición de la fuente i en la dimensión j.

Además, a cada fuente se le asigna una variable llamada **limit**, que inicialmente comienza con el valor 0. Esta variable se utiliza para definir cuántos intentos se han realizado para mejorar la fuente sin éxito. Si este valor supera un valor previamente definido, la fuente es reemplazada.

**Operaciones de abejas empleadas.**

En el algoritmo ABC, cada **abeja empleada** está asignada a una fuente de alimento, cuya posición en el espacio de búsqueda representa una solución candidata al problema de optimización. Estas abejas exploran la vecindad de la fuente asignada en busca de nuevas soluciones mejores, basándose en la calidad de la fuente (valor de fitness). Si encuentran una fuente de mayor calidad, se mueven hacia ella; de lo contrario, incrementan el valor del contador **limit**, que indica cuántos intentos han hecho sin éxito. El pseudocódigo de este tipo de abejas sería:

**Pseudocódigo de las Abejas Empleadas:**

**Para cada abeja empleada** i:

Obtener una **fuente candidata** vi​ próxima a la fuente actual xi

**Si** fitness(vi) < fitness(xi):

Moverse a la **nueva fuente** vi​.

**Sino**:

limit = limit + 1

Fin-si

Fin-para

Estas abejas para elegir la fuente candidata próxima, también llamada fuente vecina (vi), siguen esta ecuación:

Ec. 19

Donde:

: Fuente vecina elegida al azar con k ≠ i (Fuente distinta a la actual)

: número aleatorio en [-1,1]

En cada iteración, las abejas empleadas generan una nueva fuente candidata vi​, desplazándose desde la posición xi ​ hacia la fuente vecina xk​, ajustada por el valor aleatorio αi​. Si esta nueva fuente mejora la solución actual, la abeja se moverá hacia la fuente candidata.

**Operaciones de abejas observadoras.**

Las **abejas observadoras** eligen las fuentes de alimento que están siendo explotadas por las abejas empleadas, basándose en la calidad o **fitness** de dichas fuentes. La selección de la fuente se realiza en función de una **probabilidad** que refleja la calidad relativa de cada fuente en comparación con el resto. Una vez que una abeja observadora ha seleccionado una fuente, procede de manera similar a una abeja empleada, buscando una solución en la vecindad de la fuente seleccionada.

**Pseudocódigo de las Abejas Observadoras:**

**Para cada abeja observadora** i:

Elegir una fuente de alimento xi para explotar, en base a una probabilidad **pi** ([Ec. 20](#ec3Abejas))

Obtener una fuente candidata vi próxima a la fuente actual xi.

Si fitness(vi) < fitness(xi):

Moverse a la nueva fuente vi.

Sino

limiti=limiti+1

Fin-si

Fin-para

**Cálculo de la Probabilidad de Selección**

La selección de la fuente de alimento por parte de las abejas observadoras se realiza utilizando una **probabilidad de selección** ​, que está determinada por la calidad (fitness) de cada fuente en comparación con el resto de las fuentes. Esta probabilidad se calcula con la siguiente fórmula:

Ec.20

: valor de fitness de la fuente i-ésima

: contador con valores de 1 y el número de fuentes.

Las abejas observadoras eligen fuentes de alimento de manera probabilística, de acuerdo con la calidad relativa de cada fuente. Fuentes con un mayor valor de fitness tienen una probabilidad más alta de ser seleccionadas.

Después de seleccionar una fuente, esta abeja busca una fuente candidata vi cercana utilizando la [Ec. 19](#ec2Abejas)

Si la fuente candidata mejora la solución actual, la abeja observadora se mueve hacia ella, de lo contrario, incrementa el contador limiti​ asociado a la fuente actual.

**Operaciones de abejas exploradoras.**

Cuando una fuente de alimento se agota, es decir, no se ha podido mejorar después de un número determinado de intentos, la fuente se abandona y es reemplazada por una nueva fuente generada aleatoriamente. En esta fase, la abeja empleada asignada a esa fuente agotada se convierte en una abeja exploradora y busca una nueva fuente de alimento en el espacio de búsqueda. Esta nueva fuente reemplaza a la fuente agotada.

**Para cada abeja empleada i**:

**Si** limiti ​ (número de intentos fallidos) supera el valor máximo permitido:

Generar una **nueva fuente de alimento** vi​, asignando una nueva posición aleatoria en el espacio de búsqueda. [Ec.18](#ec1Abejas)

Reemplazar la fuente actual xi​ por la nueva fuente vi​.

limiti =0

Fin-si

**Fin-para**

**Mecanismo de Exploración**

* La **abeja empleada** asociada a la fuente agotada se convierte en **abeja exploradora**, y su objetivo es encontrar una **nueva fuente** aleatoria en el espacio de búsqueda.
* Una vez que se encuentra una nueva fuente vi​, la abeja empleada olvida la fuente agotada xi​ y comienza a explotar la nueva fuente vi​.
* El valor del contador limiti​​, que se utiliza para rastrear el número de intentos fallidos de mejora, se reinicializa a 0.

# 5. Tecnologías empleadas

En este apartado se describirán las tecnologías que se han utilizado para el desarrollo de este proyecto.

## 5.1 Python

Lenguaje de alto nivel elegido para la implementación de los algoritmos de optimización por enjambre y procesamiento de imágenes debido a su facilidad de uso, su rica colección de bibliotecas científicas, y su soporte para la creación de entornos virtuales.

Fue creado a finales de los años ochenta por Guido van Rossum en Stichting Mathematisch Centrum (CWI),​ en Países Bajos. Fue creado como sucesor del lenguaje ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba.[11], [12]

Es un lenguaje interpretado y de propósito general, conocido por su simpleza y legibilidad, lo que lo convierte en una opción popular tanto para principiantes como para desarrolladores experimentados. Se utiliza en una gran variedad de ámbitos, desde el desarrollo web hasta ciencia de datos, inteligencia artificial, blockchain …

Ilustración 5 Logo Python [21]

## 5.2 Numphy

NumPy es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos.

Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.

La ventaja de Numpy frente a las listas predefinidas en Python es que el procesamiento de los arrays se realiza mucho más rápido (hasta 50 veces más) que las listas, lo cual la hace ideal para el procesamiento de vectores y matrices de grandes dimensiones.[13]

Ilustración 6 Logo Numpy[22]

## 5.3 OpenCv

OpenCV es una librería de computación visual para el procesamiento de imágenes en Python. Esta biblioteca proporciona herramientas para realizar operaciones de procesamiento de imágenes, como el filtrado, la detección de bordes, el reconocimiento de características, el seguimiento de objetos, etc. Estas herramientas nos permiten desarrollar aplicaciones de visión artificial, como el reconocimiento facial, el seguimiento de objetos, etc. [14]

Esta librería ha sido muy útil en este proyecto debido a que permite el procesamiento y manipulación de imágenes. Es fundamental para la cuantificación de color en imágenes.

## 5.4 Scikit-learn

Es una biblioteca de Python que proporciona acceso a versiones eficaces de muchos algoritmos comunes. También proporciona una API propia y estandarizada. Por tanto, una de las grandes ventajas de Scikit-Learn es que una vez que se entiende el uso básico y su sintaxis para un tipo de modelo, cambiar a un nuevo modelo o algoritmo es muy sencillo. La biblioteca no solo permite hacer el modelado, sino que también puede garantizar los pasos de preprocesamiento.

La gran variedad de algoritmos y utilidades de Scikit-learn la convierten en la herramienta básica para empezar a programar y estructurar los sistemas de análisis datos y modelado estadístico. Los algoritmos de Scikit-Learn se combinan y depuran con otras estructuras de datos y aplicaciones externas como Pandas o PyBrain.

La ventaja de la programación en Python, y Scikit-Learn en concreto, es la variedad de módulos y algoritmos que facilitan el aprendizaje y trabajo del científico de datos en las primeras fases de su desarrollo.[15]

## 5.5 Scikit-image

Esta es otra librería de procesamiento de imágenes de código abierto que está diseñada para operar con las bibliotecas numéricas y científicas de Python como Numpy o SciPy.

El proyecto scikit-image comenzó como scikits.image, de Stéfan van der Walt. Su nombre proviene de la idea de que es un "SciKit" (SciPy Toolkit), una extensión de terceros desarrollada y distribuida por separado para SciPy. El código base original fue posteriormente reescrito en profundidad por otros desarrolladores. De los diversos scikits, scikit-image y scikit-learn fueron descritos como "bien mantenidos y populares" en noviembre de 2012. [16]

## 5.6 Entornos virtuales

Esta característica me ha permitido aislar las dependencias del proyecto, asegurando que las versiones correctas de las librerías se utilicen sin conflictos con otros proyectos.

Este entorno virtual es un “mini-Python” encapsulado dentro de la carpeta del proyecto, al tener su propio interprete y su propio directorio de paquetes donde se instalan las librerías hace que no afecten ni al Python global ni a otros proyectos en los que se pueda estar trabajando. También garantiza la reproducibilidad, es decir, cualquier colaborador o persona que clone el repositorio y configure el entorno ejecutará las mismas versiones de las librerías usadas.

Gracias al requirements.txt, una vez creado el entorno usando el comando **python -m venv ‘nombre’** solo tendrá que ejecutar un **pip install -r requirements.txt** para instalar las dependencias. Una vez instaladas se debe activar el entorno virtual creado con el comando **source nombreEntorno/bin/actívate** y a continuación ejecutar la herramienta como cualquier archivo Python.

## 5.7 Control de versiones

Para el desarrollo del proyecto se ha utilizado en todo momento la herramienta de GitHub lo que me ha permitido tener un control visual de los cambios que realizaba en el código y poder probar en diferentes entornos de trabajo con rapidez.

Ilustración 7 Logo GitHub[23]

## 5.8 Visual Studio Code

Ha sido el editor de código que he usado en todo el proyecto, permitiéndome la integración de los entornos virtuales de Python y la depuración del código.

Captura de pantalla de computadora

Descripción generada automáticamenteIcono

Descripción generada automáticamenteTambién me facilitó una manera de pasarle diferentes argumentos requeridos por el programa desde el propio editor formando un JSON con todos ellos.

Ilustración 8 JSON de argumentos

Ilustración 9 Logo VisualStudioCode [24]

## 5.9 Otros

Aparte de todo lo anterior se realizó un script en bash para automatizar el proceso de ejecución de los algoritmos y la generación de resultados. Dicho script lanza el programa con diferentes argumentos y vuelca los resultados en diferentes txt para recuperar los datos para formar los resultados. En estas sucesivas llamadas al programa se cambia el algoritmo que se usa, la función con la que se calcula el fitness y también las imágenes que procesa.

# 6. Pruebas

En este apartado se exponen cómo se han realizado las pruebas de la herramienta desarrollada.

Se parte de que la valoración de la imagen resultante es objetiva, es decir se calcula un valor de error basado en una fórmula matemática.

Para esta herramienta se usan índices que utilizan una imagen de referencia (Full-reference indices) ya que se tiene la imagen original, aunque también existen índices que solo usan la foto resultante.

Los índices utilizados se describen a continuación.

## 6.1 MSE

El MSE o error cuadrático medio mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se está estimando. Su ecuación es la siguiente:

Ec.20

Dónde:

es el número total de datos (pixeles, en el caso de imágenes).

es el valor real del dato .

​ es el valor predicho para el dato .

## 6.2 MAE

El MAE o error absoluto medio es una medida de la diferencia entre dos variables continuas. A continuación, se muestra su ecuación:

Ec.21

Dónde:

es el número total de datos (pixeles, en el caso de imágenes).

es el valor real del dato .

​ es el valor predicho para el dato .

## 6.3 SSIM

El SSIM es una métrica diseñada para medir la similitud entre dos imágenes. A diferencia de otras métricas como el MSE o el PSNR, que simplemente cuantifican las diferencias absolutas entre los valores de píxeles, el SSIM se basa en el modo en que los humanos perciben las diferencias estructurales entre las imágenes.

Su ecuación es la siguiente:

Ec. 22

Dónde:

y ​ son las medias de las imágenes e , respectivamente.

​ y ​ son las varianzas de e .

es la covarianza entre e .

​ y son constantes para evitar divisiones por cero.

## 6.4 UQI

El UQI (Universal Quality Index) propuesto por Wang & Bovik, evalúa la calidad reconstruida comparando tres atributos que la vista humana identifica como “lo mismo” cuando observa dos imágenes, divide la comparación entre imagen original y cuantizada en tres comparaciones: luminancia, contraste y comparaciones estructurales.

Su fórmula es la que sigue:

Ec. 23

Dónde:

denotan los valores medios de las imágenes originales y cuantizadas.

denota la desviación estándar de las imágenes originales y cuantizadas.

[17]

## 6.5 Criterios y conjuntos de pruebas

El trabajo ha sido realizado en un equipo portátil **SLIMBOOK** que cuenta con las siguientes especificaciones técnicas:

* **Procesador:** Intel Core i7-13700H (13.ª generación), con 14 núcleos / 20 hilos.
* **Memoria RAM:** 32 GB.
* **Tarjeta gráfica:** Gráficos integrados Intel (controlador Mesa Intel Graphics).

El equipo ejecuta un sistema operativo **GNU/Linux Debian**.

Se han realizado un total de veinte pruebas por cada imagen del conjunto CQ-100 para cada configuración ALGORITMO\_FUNCION y para cada tamaño de paleta.

En total se han hecho pruebas para los 5 algoritmos previamente mencionados y para las 4 funciones objetivos mencionadas. En total se han hecho las siguientes pruebas:

Siguiendo estos criterios nos da un conjunto de 160000 pruebas independientes.

Como se mencionó anteriormente, para realizar las pruebas se ha utilizado el conjunto CQ-100 propuesto por la tutora de este proyecto [18], que incluye 100 imágenes almacenadas en formato PPM.

## 6.6 Análisis

En este punto se muestran la media, denotada por a (average en inglés)y la desviación típica denotada por d (standard deviation)de los resultados obtenidos por cada configuración que se ha probado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N |  | PSO\_MAE | PSO\_MSE | PSO\_SSIM | PSO\_UQI |
| 32 | a | 7,08 | 8,38 | 20,64 | 31,21 |
|  | **d** | **0,71** | **0,77** | **3,88** | **1,99** |
| 64 | a | 8,88 | 10,32 | 21,69 | 33,1 |
|  | **d** | **0,73** | **0,75** | **3,46** | **1,81** |
| 128 | a | 12,29 | 14,22 | 25,44 | 36,75 |
|  | **d** | **0,67** | **12,42** | **3,38** | **1,82** |
| 256 | a | 19,33 | 20,38 | 32,41 | 44,55 |
|  | **d** | **1,4** | **0,96** | **3,42** | **1,76** |

Tabla 1 Tiempos PSO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N |  | FA\_MAE | FA\_MSE | FA\_SSIM | FA\_UQI |
| 32 | a | 7,03 | 8,38 | 19,85 | 31,6 |
|  | **d** | **0,71** | **0,77** | **3,38** | **2,11** |
| 64 | a | 8,86 | 10,42 | 21,89 | 33,44 |
|  | **d** | **0,7** | **2,15** | **3,58** | **1,91** |
| 128 | a | 12,53 | 13,7 | 25,66 | 37 |
|  | **d** | **0,86** | **0,76** | **3,78** | **1,83** |
| 256 | a | 19,2 | 20,48 | 32,42 | 44,93 |
|  | **d** | **1,1** | **0,93** | **3,54** | **1,83** |

Tabla 2 Tiempos FA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N |  | ABA\_MAE | ABA\_MSE | ABA\_SSIM | ABA\_UQI |
| 32 | a | 130,46 | 166,46 | 378,43 |  |
|  | **d** | **6,62** | **14,68** | **46,21** |  |
| 64 | a | 170,93 | 205,77 | 422,97 |  |
|  | **d** | **9,85** | **24,28** | **9,33** |  |
| 128 | a | 247,48 | 278,84 | 501,44 |  |
|  | **d** | **37,83** | **21,01** | **10,82** |  |
| 256 | a | 407,83 | 440,52 |  |  |
|  | **d** | **15,82** | **206,15** |  |  |

Tabla 3 Tiempos ABA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N |  | GWO\_MAE | GWO\_MSE | GWO\_SSIM | GWO\_UQI |
| 32 | a | 6,97 | 8,48 | 20,39 | 30,39 |
|  | **d** | **0,66** | **0,78** | **3,64** | **1,59** |
| 64 | a | 8,83 | 10,36 | 21,91 | 32,66 |
|  | **d** | **0,69** | **0,82** | **3,59** | **1,64** |
| 128 | a | 12,39 | 13,71 | 25,55 | 37,61 |
|  | **d** | **0,77** | **0,73** | **3,67** | **8,33** |
| 256 | a | 19,01 | 20,5 | 32,19 | 43,99 |
|  | **d** | **0,94** | **0,9** | **3,44** | **1,96** |

Tabla 4 Tiempos GWO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N |  | WOA\_MAE | WOA\_MSE | WOA\_SSIM | WOA\_UQI |
| 32 | a | 7,06 | 8,5 | 20,5 | 30,97 |
|  | **d** | **0,67** | **0,79** | **3,59** | **1,82** |
| 64 | a | 8,92 | 10,19 | 22,18 | 33,41 |
|  | **d** | **0,76** | **0,69** | **3,58** | **8,11** |
| 128 | a | 12,36 | 13,96 | 25,43 | 36,42 |
|  | **d** | **0,67** | **0,88** | **3,57** | **1,76** |
| 256 | a | 19,16 | 20,75 | 32,63 | 43,93 |
|  | **d** | **0,85** | **1,17** | **3,66** | **2,04** |

Tabla 5 Tiempos WOA

Los tiempos de todas las pruebas están medidos en segundos. Como se puede observar en las tablas 1, 2, 3, 4 y 5 el algoritmo más rápido en ejecutarse fue el WOA con el índice de error MAE por una ligera ventaja. El índice que menos tarda en cualquier caso siempre es el MAE debido a que es una métrica muy barata de evaluar.

Otra conclusión a la que podemos llegar es que el algoritmo mas lento es el ABA sin ninguna duda ya que existe una gran diferencia con el resto de los algoritmos, también podemos observar que, aunque este algoritmo sea el mas lento, se repite la conclusión de que el índice MAE es el más rápido de todos. Aun así el ABA triplica el tiempo del resto de los algoritmos, lo que indica que el propio mecanismo de exploración/explotación de la colonia añade un coste considerable.

En cuanto a la paleta elegida podemos observar que el tiempo aumenta llegando a triplicar o cuadruplicar el tiempo si se escoge una paleta de mayor tamaño.

Por último, podemos observar que el SSIM o el UQI aumentan el tiempo de ejecución considerablemente, esto es debido a la manera de calcular estos índices ya que el SSIM o el UQI obligan a calcular medias, varianzas y covarianzas, mientras que el MAE o el MSE se resuelven con operaciones elementales.

# 7. Desarrollo del programa

Para el desarrollo como se ha mencionado antes se realizó un programa en Python que realiza las operaciones de los algoritmos de enjambre y genera imágenes cuantizadas con las que calcula un error con 4 índices de error diferentes (MAE, MSE, SSIM, UQI), este programa se llama con diferentes argumentos que se explicarán en este apartado.

Por otro lado, también se realizó un script para la automatización de la obtención de resultados que se encarga de llamar al programa repetidas veces para ejecutarlo con todos los algoritmos, todas las funciones y 4 paletas de colores diferentes, estas 4 paletas son de 32 colores, de 64, de 128 y de 256.

## 7.1 Archivo main o ejecutor

Este archivo es el principal del programa encargado de recibir los argumentos que se le pasen y de elegir o “mapear” estos argumentos con el algoritmo que toque junto con la función que toque.

Para recibir estos argumentos se ha utilizado la librería por defecto de Python llamada argparse.

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Ilustración 10 Argumentos

En la ilustración 10 se ve como se añaden los argumentos de tipo cadena o de tipo entero, estos argumentos por orden son:

* Imagen: imagen que se va a procesar en el algoritmo.
* NumeroColores: el número de colores de la nueva imagen o la paleta que se va a usar.
* Algoritmo: algoritmo que usará el programa para la cuantificación de color. Estos pueden ser: PSO, FA, BA, GWO, ABA que respectivamente son: pso, luciérnagas, ballenas, lobos y abejas.
* Función: función que calculará el fitness de cada iteración del algoritmo, las opciones son: MSE, MAE, SSIM, MSSIM.
* Iteraciones: número entero que representa el número de iteraciones que realizara el algoritmo escogido.
* Individuos: número entero que representa la población de individuos que realizarán el algoritmo.
* PintaImagen: este argumento booleano solo lo usé para mis pruebas, si viene a True el programa mostrará por pantalla las dos imágenes, tanto la original como la resultante, me fue útil al principio para comprobar visualmente que la imagen resultante realmente estaba modificada.
* a: este último argumento indica si se ajusta la paleta generada. Si recibe el valor de 1 se ajusta.

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Ilustración 11 Obtener ruta de los directorios de imágenes

En la ilustración 11 se ve la línea que obtiene la ruta de la imagen que se le ha pasado así como del directorio donde se almacenan las imágenes cuantizadas. Se ha usado el módulo os para obtenerla de manera que funcione bien en cualquier sistema operativo. Esta operación se realiza debido a que las imágenes que se procesan en el algoritmo están en una carpeta imagenes que está ubicada al mismo nivel que el archivo ejecutor, si estuviesen al mismo nivel que ejecutor no haría falta, pero esto sería más caótico. args.imagen es la forma de obtener la imagen que se le ha pasado como argumento.

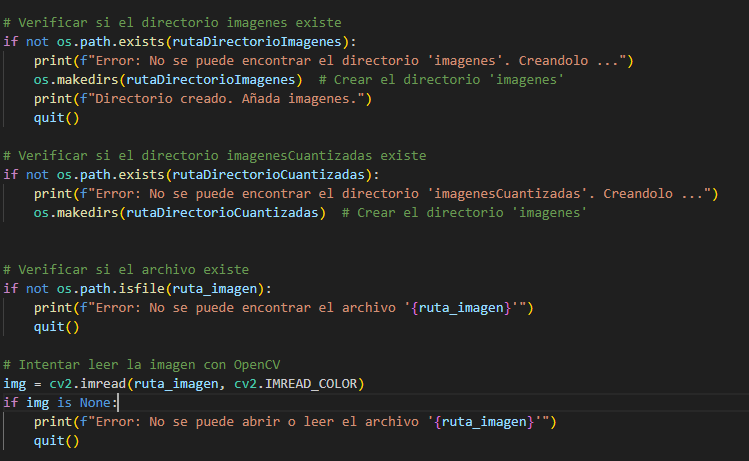
En la ilustración 12 se ven algunas comprobaciones necesarias por si no se pudiese encontrar el archivo en la carpeta images y la comprobación de que se puede leer o abrir la imagen pasada.

Ilustración 12 Comprobaciones de rutas e imágenes.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 13 Comprobaciones extra

En la ilustración 13 se muestran las comprobaciones del resto de argumentos. Comprueba si se ha recibido un numero válido de iteraciones, colores e individuos.

También comprueba si el algoritmo y la función recibidos están dentro de los valores permitidos

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Ilustración 14Diccionarios Python de algoritmos y funciones

En la ilustración 14 se muestran los diccionarios Python que hacen referencia a las funciones y algoritmos implementados. Estos diccionarios nos permiten acceder a las diferentes configuraciones de la herramienta, también nos simplifica el desarrollo de nuevas funciones y algoritmos en un futuro, ya que lo único que se debe hacer es añadirlo a estos diccionarios de la misma manera que están estos.

En el diccionario de las funciones objetivo se debe poner el valor que debe recibir la herramienta como una cadena y el valor real que tiene, es decir, el método que hace referencia a la función objetivo desarrollado en el módulo de funciones.

En el diccionario de algoritmos se debe añadir el calor que recibe la herramienta como una cadena y la función lambda a la que hace referencia, es decir, la referencia al constructor del algoritmo que se esté implementando.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Ilustración 15 Método ejecutor de la herramienta

En la ilustración 15 se muestra el método que llama al algoritmo indicado con la función objetivo que se le ha pasado. Realiza comprobaciones y escoge el valor pedido recuperándolo de los diccionarios que se mencionaron anteriormente.

Todas estas operaciones constituyen la primera parte del programa, donde se reciben argumentos, se comprueban y se “escoge” el algoritmo a usar junto a la función seleccionada.

## 7.2 Script

Una vez definido como se ejecuta la herramienta se explica el script realizado para obtener todas las pruebas.

Texto

El contenido generado por IA puede ser incorrecto.Como resumen, este script llama repetidas veces a la herramienta ejecutando una serie de bucles que componen todas y cada una de las posibles configuraciones que puede realizar la herramienta para componer todos los resultados.

Ilustración 16 Constantes y directorios

En la anterior ilustración se muestran las constantes del script, todos los algoritmos se ejecutaron con 15 iteraciones y 20 individuos, También se pueden ver los directorios, la propia ruta donde está el script, el directorio de checkpoints, y los directorios auxiliares tanto de las imágenes cuantizadas como el directorio donde se ubica el piq-master que es la herramienta usada para calcular los diferentes índices de error con los que se han generado los resultados [19].

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Ilustración 17 Bucles

En la ilustración 17 se muestran los bucles que componen las diferentes configuraciones de la herramienta en cuanto al algoritmo, función, paleta y ajuste requeridos.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 18 Sistema de checkpoints

Debido al tiempo necesario para ejecutar todas las pruebas de este trabajo, tuve que implementar un pequeño sistema de checkpoints que me permitieran guardar los puntos que ya tenia probados. Este sistema se divide en comprobaciones tanto para conocer si una configuración ya está ejecutada, es decir, si se ha ejecutado el PSO con la función MAE con una paleta de 32 colores, por ejemplo, y la comprobación de la imagen que se ha probado.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Ilustración 19 Bucle Imagenes

En la ilustración 19 se muestra el bucle de las imágenes del conjunto CQ-100 y la comprobación de si ya se ha procesado.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 20 Comienzo de la prueba

En esta ilustración se muestra el mensaje que se escribe por la terminal para conocer el estado en el que se encuentra el script, y un bucle que hace que cada imagen se ejecute 20 veces para generar el banco de datos.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 21 Ejecución de la herramienta

En la ilustración 21 se muestra la ejecución de la herramienta llamando a ejecutor.py, y el cálculo de los índices de error llamando a errores\_cq.py donde se calculan algunos de ellos y a image\_metrics.py que es del paquete piq-master donde se calcula el resto de ellos. También se captura el momento en el que empezó a ejecutarse para posteriormente poder conocer el tiempo que se tardo en ejecutar, el cual se escribe al final de cada línea en los txt de resultados.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 22 Fin del script

En esta ilustración se muestra el borrado de la imagen cuantizada después de calcular los índices de error y la actualización del archivo de checkpoints.

# 8. Trabajo futuro

Existen varias áreas en las que este trabajo podría expandirse y mejorar, algunos de ellos son:

* **Optimización de Algoritmos:** Aunque los algoritmos de enjambre como PSO, GWO, y ABA han sido aplicados exitosamente, se podrían explorar versiones avanzadas de estos algoritmos, como versiones híbridas o adaptativas que ajusten sus parámetros dinámicamente durante la ejecución. Esto podría mejorar la convergencia y reducir el tiempo de procesamiento.
* **Ampliación de la Biblioteca de Algoritmos:** Actualmente, la herramienta incluye un conjunto de algoritmos específicos. Una dirección interesante sería la incorporación de nuevos algoritmos de optimización basados en la naturaleza, como el algoritmo de murciélagos, el algoritmo de colonias de hormigas o variantes de redes neuronales evolutivas, ampliando las opciones para la cuantificación de color.
* **Paralelización y Uso de GPU:** A medida que las imágenes procesadas son de mayor tamaño o mayor resolución, los tiempos de cómputo aumentan considerablemente. La paralelización del código o el uso de procesamiento en GPU podría reducir de manera significativa el tiempo de ejecución de los algoritmos, haciéndolos más eficientes en escenarios de grandes volúmenes de datos.
* **Evaluación de Nuevas Métricas:** Las métricas de calidad utilizadas, como el MSE, MAE, SSIM y MS-SSIM, ofrecen una visión del rendimiento, pero podrían incluirse nuevas métricas que tomen en cuenta aspectos específicos de la percepción humana del color. Métricas más sofisticadas que evalúen la fidelidad del color y las características visuales serían un buen complemento.

# 9. Conclusiones

El desarrollo de esta herramienta para la aplicación de algoritmos de enjambre en la cuantificación de color ha demostrado la efectividad y versatilidad de estas técnicas bioinspiradas en el ámbito del procesamiento de imágenes. A lo largo del proyecto, se han implementado y probado diversos algoritmos de optimización como PSO (Optimización por Enjambre de Partículas), GWO (Optimización por Lobos Grises), FA (Luciérnagas), y ABA (Abejas Artificiales). Estos métodos han mostrado su capacidad para reducir el número de colores en una imagen manteniendo un alto nivel de calidad visual, lo que los convierte en herramientas útiles para aplicaciones de compresión de imágenes, procesamiento gráfico y diseño visual.

La principal ventaja de los algoritmos de enjambre reside en su capacidad para explorar eficientemente el espacio de búsqueda y encontrar soluciones óptimas en problemas complejos, como la cuantificación de color. Además, estos algoritmos son altamente flexibles, permitiendo su ajuste y personalización para adaptarse a diferentes escenarios y requisitos de precisión.

En resumen, este trabajo ha evidenciado el potencial de los algoritmos de enjambre en el ámbito del procesamiento de imágenes, y la herramienta desarrollada servirá como una base sólida para futuras investigaciones y aplicaciones en este campo.

# 10.Bibliografía

[1] «Los modos de color de la imagen digital». Accedido: 28 de enero de 2024. [En línea]. Disponible en: https://www.fotonostra.com/fotografia/modoscolor.htm

[2] «Inteligencia de enjambre e inteligencia artificial - Fundación Aquae». Accedido: 5 de febrero de 2024. [En línea]. Disponible en: https://www.fundacionaquae.org/la-inteligencia-enjambre-y-la-inteligencia-artificial/

[3] «PSO: Optimización por enjambres de partículas». Accedido: 21 de abril de 2024. [En línea]. Disponible en: https://www.cs.us.es/~fsancho/Blog/posts/PSO.md

[4] «Algoritmo firefly - Wikipedia, la enciclopedia libre». Accedido: 5 de mayo de 2024. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Algoritmo\_firefly

[5] S. Mirjalili, S. M. Mirjalili, y A. Lewis, «Grey Wolf Optimizer», *Advances in Engineering Software*, vol. 69, pp. 46-61, mar. 2014, doi: 10.1016/J.ADVENGSOFT.2013.12.007.

[6] «GWO». Accedido: 20 de mayo de 2024. [En línea]. Disponible en: https://seyedalimirjalili.com/gwo

[7] H. Faris, I. Aljarah, M. A. Al-Betar, y S. Mirjalili, «Grey wolf optimizer: a review of recent variants and applications», *Neural Comput Appl*, vol. 30, n.o 2, pp. 413-435, jul. 2018, doi: 10.1007/S00521-017-3272-5/TABLES/4.

[8] J. Nasiri y F. M. Khiyabani, «A whale optimization algorithm (WOA) approach for clustering», *Cogent Math Stat*, vol. 5, n.o 1, p. 1483565, ene. 2018, doi: 10.1080/25742558.2018.1483565.

[9] «AN IDEA BASED ON HONEY BEE SWARM FOR NUMERICAL OPTIMIZATION».

[10] «Algoritmo colonia de abejas artificiales - Wikipedia, la enciclopedia libre». Accedido: 4 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Algoritmo\_colonia\_de\_abejas\_artificiales

[11] *Why was Python created in the first place?* General Python FAQ. Accedido: 4 de septiembre de 2024. [En línea]. Disponible en: http://docs.python.org/faq/general#why-was-python-created-in-the-first-place

[12] «artima - The Making of Python», *www.artima.com*, Accedido: 4 de septiembre de 2024. [En línea]. Disponible en: https://www.artima.com/articles/the-making-of-python

[13] «La librería Numpy | Aprende con Alf». Accedido: 4 de septiembre de 2024. [En línea]. Disponible en: https://aprendeconalf.es/docencia/python/manual/numpy/

[14] «Procesamiento de imágenes con OpenCV en Python». Accedido: 4 de septiembre de 2024. [En línea]. Disponible en: https://imaginaformacion.com/tutoriales/opencv-en-python

[15] «Scikit-Learn, herramienta básica para el Data Science en Python». Accedido: 4 de septiembre de 2024. [En línea]. Disponible en: https://www.master-data-scientist.com/scikit-learn-data-science/

[16] S. Van Der Walt *et al.*, «scikit-image: image processing in Python», *PeerJ*, vol. 2:e453, n.o 1, p. e453, jul. 2014, doi: 10.7717/peerj.453.

[17] A. León-Batallas, J. Bermeo-Paucar, J. Paredes-Quevedo, y H. Torres-Ordoñez, «Una revisión de las métricas aplicadas en el procesamiento  de imágenes», *RECIMUNDO*, vol. 4, n.o 3, pp. 267-273, sep. 2020, doi: 10.26820/recimundo/4.(3).julio.2020.267-273.

[18] M. E. Celebi y M.-L. Pérez-Delgado, «CQ100: A High-Quality Image Dataset for Color Quantization Research», vol. 2, 2022, doi: 10.17632/VW5YS9HFXW.2.

[19] «GitHub - photosynthesis-team/piq: Measures and metrics for image2image tasks. PyTorch.» Accedido: 15 de junio de 2025. [En línea]. Disponible en: https://github.com/photosynthesis-team/piq

[20] «Así son los LOBOS GRISES ¡Impresionantes! - Impacto Venezuela». Accedido: 15 de junio de 2025. [En línea]. Disponible en: https://impactove.com/asi-son-los-lobos-grises-impresionantes/

[21] «Python logo - Iconos Social Media y Logos». Accedido: 15 de junio de 2025. [En línea]. Disponible en: https://icon-icons.com/es/icono/python-logo/168886

[22] «File:NumPy logo 2020.svg - Wikipedia». Accedido: 15 de junio de 2025. [En línea]. Disponible en: https://en.m.wikipedia.org/wiki/File:NumPy\_logo\_2020.svg

[23] «1000 Logos - The Famous logos and Popular company logos meaning». Accedido: 15 de junio de 2025. [En línea]. Disponible en: https://1000logos.net/

[24] «Visual Studio Code Logo PNG vector in SVG, PDF, AI, CDR format». Accedido: 15 de junio de 2025. [En línea]. Disponible en: https://logowik.com/visual-studio-code-vector-logo-1-5273.html