Índice

[1. Introducción 3](#_Toc166868111)

[1.1 Representación de imágenes en color 4](#_Toc166868112)

[1.2 Los modos de color 5](#_Toc166868113)

[1.3 El espacio de color RGB 6](#_Toc166868114)

[2. El problema de la cuantificación de color 8](#_Toc166868115)

[2.1 Definición del problema 8](#_Toc166868116)

[2.2 Aplicaciones prácticas 9](#_Toc166868117)

[3. Los algoritmos de enjambres. 11](#_Toc166868118)

[3.1 Características generales. 11](#_Toc166868119)

[3.2 Algoritmo de optimización por enjambre de partículas (PSO) 13](#_Toc166868120)

[3.2.1 Introducción 13](#_Toc166868121)

[3.2.2 Autores 14](#_Toc166868122)

[3.2.3 Variables 15](#_Toc166868123)

[3.2.4 Definición del algoritmo 15](#_Toc166868124)

[3.2.4 Métodos de cálculo para la inercia (w) 18](#_Toc166868125)

[3.2.5 Parámetros PSO 20](#_Toc166868126)

[3.3 Luciérnagas (Firefly) 22](#_Toc166868127)

[3.3.1 Introducción 22](#_Toc166868128)

[3.3.2 Autor 22](#_Toc166868129)

[3.3.3 Variables 23](#_Toc166868130)

[3.3.4 Definición del algoritmo 24](#_Toc166868131)

[3.4 Lobos / Grey Wolf optimization (GWO) 27](#_Toc166868132)

[3.4.1 Introducción 27](#_Toc166868133)

[3.4.2 Fundamentos 28](#_Toc166868134)

[3.4.3 Definición del algoritmo 30](#_Toc166868135)

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

## 1. Introducción

La reducción de color en imágenes es un problema complejo en estos tiempos en los que nos comunicamos frecuentemente con el uso de imágenes ya que involucra la transformación de una imagen con una gran cantidad de colores en una versión simplificada con un número limitado de colores.

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 resultados óptimos. Algunas de las razones que explican por qué la reducción de color en imágenes es un problema complejo son:

* **Gran volumen de datos:** Ya que estas imágenes suelen contener millones de píxeles y cada píxel tiene diversa información sobre sí mismo, 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 reducida de manera precisa es una tarea compleja. La selección de estos colores debe hacerse de manera inteligente para garantizar que la imagen simplificada 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 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 una imagen reducida 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.

## 1.1 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 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, 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.

## 1.2 Los modos de color

Ya que se ha hablado de como un ordenador representa las imágenes en color utilizando el modo de color o modelo 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 modo 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 modo compone los colores de manera sustractiva, es decir, parte del blanco y va restando colores usando sus complementarios.

Este modo 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 modo 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 (como 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]

## 1.3 El espacio de color RGB

Un espacio de color es un sistema de interpretación del color, es decir, una organización específica de los colores en una imagen o video. Depende del modelo de color en combinación con los dispositivos físicos que permiten las representaciones reproducibles de color, por ejemplo, las que se aplican en señales analógicas (televisión a color) o representaciones digitales. Un espacio de color puede ser arbitrario, con colores particulares asignados según el sistema y estructurados matemáticamente.[2]

Los algoritmos de reducción de color utilizan varios métodos para seleccionar un conjunto limitado de colores representativos de la imagen original. Estos colores se eligen mediante métodos de agrupación o clustering como el algoritmo K-means, donde se agrupan los colores similares en un número menor de grupos, y cada grupo se representa con un color promedio.

Una vez que se han seleccionado los colores representativos, la imagen se reemplaza por una versión simplificada en la que los píxeles originales se ajustan al color más cercano en la paleta de colores reducida. Esto permite reducir la cantidad de información de color en la imagen, lo que ahorra espacio de almacenamiento y puede mejorar la eficiencia en el procesamiento y la transmisión de imágenes, entre otros beneficios.

En resumen, el espacio de color RGB es la base para representar colores en imágenes digitales, y la técnica de reducción de color utiliza estrategias basadas en este espacio para simplificar y optimizar la representación de colores en una imagen, manteniendo al mismo tiempo la calidad visual deseada. Los algoritmos de enjambres y otras técnicas de optimización pueden desempeñar un papel importante en la selección de los colores representativos en este contexto.

# 2. El problema de la cuantificación de color

## 2.1 Definición del problema

Una imagen digital se forma con una cantidad indeterminada de píxeles la cual usa un cierto espacio de color. Cuando este espacio de color es el RGB, 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 numero de colores evitando la pérdida de información. 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, como 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. 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. 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.

## 2.2 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.

# 3. Los algoritmos de enjambres.

## 3.1 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.[3]

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.

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

### 3.2.1 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 método fue descrito alrededor de 1995 por Kennedy y Eberhart, y se inspira en el comportamiento de los enjambres de insectos en la naturaleza. En concreto, podemos pensar en un enjambre de abejas, ya que éstas a la hora de recolectar polen buscan la región del espacio en la que existe más densidad de flores, porque la probabilidad de que haya polen es mayor. La misma idea fue trasladada al campo de la computación en forma de algoritmo y se emplea en la actualidad en la optimización de distintos tipos de sistemas.

Formalmente hablando, se supone que tenemos una función desconocida, f(x1,…,xn), que podemos evaluar en los puntos que queramos pero a modo de caja negra, por lo que no podemos conocer su expresión. El objetivo es el habitual en optimización, encontrar valores de x1,…,xn para los que la función f sea máxima (o mínima, o bien verifica alguna relación extrema respecto a alguna otra función). A f se le suele llamar función fitness, ya que va a determinar cómo de buena es la posición actual para cada partícula.[4]

### 3.2.2 Autores

El Algoritmo de Optimización por Enjambre de Partículas (Particle Swarm Optimization, PSO) fue desarrollado por James Kennedy, un científico social, y Russell Eberhart, un ingeniero eléctrico, en 1995. El algoritmo se inspira en el comportamiento social y de forrajeo de las aves y los peces. Kennedy y Eberhart presentaron el PSO como una técnica computacional que simula el comportamiento social de los enjambres para resolver problemas de optimización. Desde su creación, el PSO ha sido ampliamente utilizado y adaptado para una variedad de aplicaciones complejas en varios campos gracias a su simpleza y eficacia.

**James Kennedy** (nacido el 5 de noviembre de 1950) es un psicólogo social estadounidense, más conocido como creador e investigador de la optimización del enjambre de partículas. Los primeros artículos sobre el tema, de Kennedy y Russell C. Eberhart, se presentaron en 1995; Desde entonces se han publicado decenas de miles de artículos sobre enjambres de partículas. El libro Academic Press/Morgan Kaufmann, Swarm Intelligence, de Kennedy y Eberhart con Yuhui Shi, se publicó en 2001. [5]

**Russell C. Eberhart**, ingeniero eléctrico estadounidense, mejor conocido como codesarrollador del concepto de optimización del enjambre de partículas. Es profesor de Ingeniería Eléctrica e Informática y profesor adjunto de Ingeniería Biomédica en la Escuela Purdue de Ingeniería y Tecnología de la Universidad Purdue de Indiana (IUPUI). Es miembro del IEEE y miembro del Instituto Americano de Ingeniería Médica y Biológica.

Obtuvo un doctorado. en ingeniería eléctrica de la Universidad Estatal de Kansas en 1972. Fue editor asociado de IEEE Transactions on Evolutionary Computation y ex presidente del IEEE Neural Networks Council. [6]

### 3.2.3 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(t).

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.

### 3.2.4 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.

**HASTA**(condición de parada)

**Inicializar la población de individuos**

Al principio del algoritmo se inicializan las variables de cada individuo para comenzar. Para cada individuo se inicia 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 ].

**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 serán tales como MSE, MAE, SSIM o MS-SSIM (a partir de ahora función objetivo).

**fitnessi ≈ f(xi )**

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 esta.

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(t).

**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:

vi (t+1) = **w** vi (t) + **f1** **e1** [bi (t)-xi (t)] + **f2** **e2** [g(t)-xi (t)]

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:

vi (t+1) = **w vi (t)** + f1 e1 [bi (t)-xi (t)] + f2 e2 [g(t)-xi (t)]

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

-La distancia a la mejor posición personal:

vi (t+1) = w vi (t) + **f1 e1 [bi (t)-xi (t)]** + f2 e2 [g(t)-xi (t)]

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:

vi (t+1) = w vi (t) +f1 e1 [bi (t)-xi (t)] + **f2 e2 [g(t)-xi (t)]**

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: vi (t+1) deberá estar entre [vmin , vmax ].

Para ello:

- Si vi (t+1) > vmax  => vi (t+1) = vmax

-Si vi (t+1) < vmin  => vi (t+1) = vmin

1. *Actualizar posición de cada individuo.*

Se calcula la nueva posición de cada individuo i.

xi (t+1) = xi (t) + vi (t+1)

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

### 3.2.4 Métodos de cálculo para la inercia (w)

Como se ha mencionado antes, el parámetro de inercia en el PSO es recomendable que vaya decreciendo con el tiempo, ya que este parámetro se encarga de regular la velocidad asociada a cada partícula. Esta demostrado que disminuir este parámetro durante la ejecución del algoritmo mejora el resultado, aunque originalmente no se contemplaba.

En 1998, Shi y Eberhart introdujeron el concepto de masa inercial al aplicar un coeficiente de inercia para limitar la velocidad de las partículas en el algoritmo PSO; a su vez establecieron que dicho coeficiente facilitaba la búsqueda global cuando se trataba de un número grande, mientras que si se trataba de un número pequeño se facilitaba la búsqueda local.

Sin embargo, se ha demostrado en numerosos estudios que el parámetro de inercia, con ajuste dinámico en el tiempo, puede aumentar significativamente la convergencia de una solución, comparándolo con un valor constante

Para las pruebas realizadas se consideraron dos métodos de descenso de la inercia: lineal y caótico.

1.Lineal

Este método ajusta la inercia siguiendo la fórmula:

De donde wmax es el valor inicial del parámetro de inercia al iniciar el algoritmo y wmin representa el valor final. Se ha reportado que el rango idóneo para el parámetro w es de [0.9 - 0.4]. El método de descenso lineal habitualmente presenta problemas de convergencia con funciones que tiene una mayor cantidad de variables en su solución.

2-Caótico

La inercia caótica fue propuesta por Feng en donde agrega al método de descenso lineal un factor que genera un mapeo caótico en el rango [0,1]:

Donde 3.57< μ ≤4 y z es un número aleatorio de [0,1]. Sin embargo, cuando μ = 4 se cubre el intervalo de [0,1]. Se ajusta entonces con esta fórmula:

[7]

### 3.2.5 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 si 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 mas 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 cualquier individuo.

## 3.3 Luciérnagas (Firefly)

### 3.3.1 Introducción

El algoritmo firefly, es un algoritmo metaheurístico, inspirado en el comportamiento del centelleo de las luciérnagas. 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.

El algoritmo firefly es un procedimiento metaheurístico de optimización inspirado en la naturaleza.[8]

### 3.3.2 Autor

El autor de este algoritmo fue **Xin-She Yang** en 2008.

Yang es un dedicado investigador en el campo de la ingeniería y la informática, escribió sobre este algoritmo al intentar utilizar diversos métodos inspirados en la naturaleza para resolver problemas que son difíciles de resolver usando técnicas de optimización convencionales.[9]

Xin-She Yang es un matemático británico de origen chino, investigador experto del Laboratorio Físico Nacional, es conocido por desarrollar varios algoritmos heurísticos para optimización en ingeniería.

Ideó el algoritmo firefly (2008), la búsqueda cuckoo​ (2009), el algoritmo de murciélago (2010),​ y el algoritmo de polinización floral.

Estos algoritmos se han convertido en herramientas importantes en inteligencia artificial,​ aprendizaje de máquinas, informática neuronal y aplicaciones de ingeniería. Desde 2009, más de 1.000 artículos científicos de publicaciones acreditadas han citado el algoritmo firefly y la búsqueda cuckoo. Además, desarrolló la hipótesis de Van Flandern-Yang en colaboración con Tom Van Flandern para explicar las variaciones de gravedad durante el eclipse solar de 1977,13​ y con otros fenómenos físicos.​ [10]

### 3.3.3 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 → fi =(xi1 , ...xir )

Brillo → Li ≈ f(fi )

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 la 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.

### 3.3.4 Definición del algoritmo

**PSEUDOCODIGO**

Generar la población inicial de luciérnagas

**REPETIR**

Mover cada luciérnaga hacia las más brillantes

Mover la luciérnaga más brillante

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 individuo (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 mas brillante que él, siguiendo esta ecuación:

fi(t+1) = fi(t) + β(rik)[fk(t) - fi(t) ] + εi

Donde:

fi(t+1) -> es la siguiente posición del individuo i

fi(t) -> es la posición actual del individuo i

β(rik) -> es el atractivo del individuo k sobre el i

ε-> un valor aleatorio

Donde β(rik) se calcula mediante:

-> 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:

fbest(t+1) = fbest(t) + ε

**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 ultimo 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). Algunos de ellos son:

* Quicksort.
* Mergesort.
* Heapsort.
* Insertion Sort ….

Como se ha dicho, esta elección depende de la implementación y entorno de ejecución. Por ejemplo, si se utiliza Python es posible usar el método de ordenamiento integrado que es TimSort bajo el método .short() de las listas, que suele ser suficiente para la mayoría de las aplicaciones prácticas.

## 3.4 Lobos / Grey Wolf optimization (GWO)

### 3.4.1 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 mas 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.[11]

### 3.4.2 Fundamentos

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

-Cada único 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. ->

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:

A = 2ar1 - a (Ec 1) C = 2r2 (Ec 2)

Donde:

r1, r2: son vectores aleatorios en [0, 1].

a: 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.

### 3.4.3 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 N 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

Ajustar el parámetro a

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 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

fi ≈ f(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:

Xp(t+1) = (X1 +X2 + X3 ) / 3

X1 = Xα - A1Dα Dα =|C1Xα - Xp(t)|

X2 = Xβ – A2Dβ Dβ =|C2Xβ – Xp (t)|

X3 = Xδ - A1Dδ Dδ =|C3Xδ – Xp (t)|

En donde:

T: es la iteración actual.

Ai , Ci (i=1, 2, 3): vectores de coeficientes, calculados usando la Ec1 y la Ec2.

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:

En donde:

t: es la iteración actual del algoritmo.

Max: es el número máximo de iteraciones del algoritmo.

[12]

[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] «Espacio de color - Wikipedia, la enciclopedia libre». Accedido: 28 de enero de 2024. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Espacio\_de\_color

[3] «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/

[4] «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

[5] «James Kennedy (social psychologist) - Wikipedia». Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://en.wikipedia.org/wiki/James\_Kennedy\_(social\_psychologist)

[6] «Russell C. Eberhart - Wikipedia». Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Russell\_C.\_Eberhart

[7] V. Álvarez-Garduño, N. Guadiana-Ramírez, y Á. Anzueto-Ríos, «Análisis comparativo de la modificación del parámetro de inercia para la mejora en el desempeño del algoritmo PSO», *Científica*, vol. 25, n.o 1, pp. 104-114, 2021, doi: 10.46842/IPN.CIEN.V25N1A09.

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

[9] «Dr Xin-She Yang | Middlesex University». Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://www.mdx.ac.uk/about-us/our-people/staff-directory/dr-xin-she-yang/

[10] «Xin-She Yang - Wikipedia, la enciclopedia libre». Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Xin-She\_Yang

[11] 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.

[12] 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.