

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/303803095>

Revisión de los Algoritmos Bioinspirados

Thesis · July 2014

CITATION

1

READS

1,707

2 authors, including:



Ignacio Riquelme Medina
Royal College of Surgeons in Ireland

12 PUBLICATIONS 68 CITATIONS

[SEE PROFILE](#)



ALGORITMOS BIOINSPIRADOS: Una revisión según sus fundamentos biológicos.

Ignacio Riquelme Medina
Julio 2014

Índice

Portada.....	
Índice.....	
Resumen.....	0
I. Introducción:.....	1
II. Descripción de los algoritmos.....	4
II.1 Algoritmos evolutivos.....	4
II.2 Algoritmos de inteligencia de enjambre.....	8
II.3 Ecología.....	17
II.4 Mixtos.....	20
III. Tabla comparativa.....	21
IV. Conclusión.....	24
V. Bibliografía.....	25

Resumen:

En las últimas décadas conforme la ciencia y la tecnología han ido avanzando, el tamaño y la complejidad de los problemas a los que nos enfrentábamos han sido cada vez mayores y con ello se hizo necesaria la búsqueda de nuevos procedimientos para hacer frente a dicha problemática. Una de estas nuevas fuentes de inspiración ha sido la naturaleza ya que es capaz de proveer soluciones sencillas y eficaces a cuestiones que son mucho más difíciles de resolver con los métodos utilizados hasta ese momento. En el caso de los problemas de optimización en las ciencias de la computación, este hecho ha dado lugar a la aparición de los algoritmos bioinspirados, que adoptan fenómenos presentes en la naturaleza para la resolución de estos problemas, mostrando resultados sorprendentes hasta el momento. Este ha sido el motivo por el que se ha realizado un gran esfuerzo en su desarrollo, dando como resultado una amplia variedad de algoritmos bioinspirados. En este trabajo se va a hacer una revisión de los algoritmos inspirados en la naturaleza más novedosos y eficaces que se han desarrollado en los últimos años, categorizados según su base biológica y su método en: algoritmos evolutivos, de enjambre, basados en ecología y mixtos. Se realizará una breve descripción de su base, los procesos que utilizan, sus principales usos y ventajas.

Palabras clave: algoritmo, bioinspirado, computación.

I. Introducción

En la ingeniería, los problemas a los que nos enfrentamos en la realidad pueden ser formulados matemáticamente en forma de problemas de optimización[1], en los que se busca encontrar la mejor solución posible al mismo, entre todas sus soluciones disponibles. Como primeras herramientas para intentar dar respuesta a este tipo de problemas se utilizaron los métodos analíticos deterministas (es decir, que no usan la probabilidad). Sin embargo, el esfuerzo computacional requerido por estos métodos aumenta excesivamente conforme aumenta la complejidad del problema, incrementándose también el número de errores y por tanto se hacen inadecuados para la resolución de los más complejos. Este es el motivo principal por el que se están buscando métodos alternativos a los deterministas que resulten más eficientes y ofrezcan buenas soluciones. Estos nuevos métodos son algoritmos de optimización estocásticos (es decir, basados en la probabilidad), siendo mucho más eficientes que los deterministas. Estos métodos pueden ser heurísticos (métodos basados en el aprendizaje y la experiencia para resolver problemas), meta-heurísticos (métodos heurísticos que pueden ser empleados de forma general al asumir ciertas variables de los problemas), basados en poblaciones, etc.

En esta búsqueda de nuevos métodos de optimización, muchos investigadores han vuelto su vista a la naturaleza, ya que muchos de los procesos que ocurren en ella pueden ser vistos como la búsqueda de soluciones a distintos problemas, es decir procesos de optimización. Para cada problema que nos encontramos en la naturaleza, ésta ha sido capaz de ofrecer soluciones eficaces, con poca o nula compresión del mismo. Además estos se resuelven a través de estrategias relativamente sencillas, por lo que son fáciles de implementar. Por ejemplo podemos encontrar procesos de adaptación al medio (como estrategias de búsqueda de alimento o pareja), interacción con otras especies (como la competencia o la depredación, tanto a nivel individual como en grupos), estrategias a nivel de ecosistemas (como la colonización) o la evolución, que puede ser considerada como un proceso de optimización en sí misma.

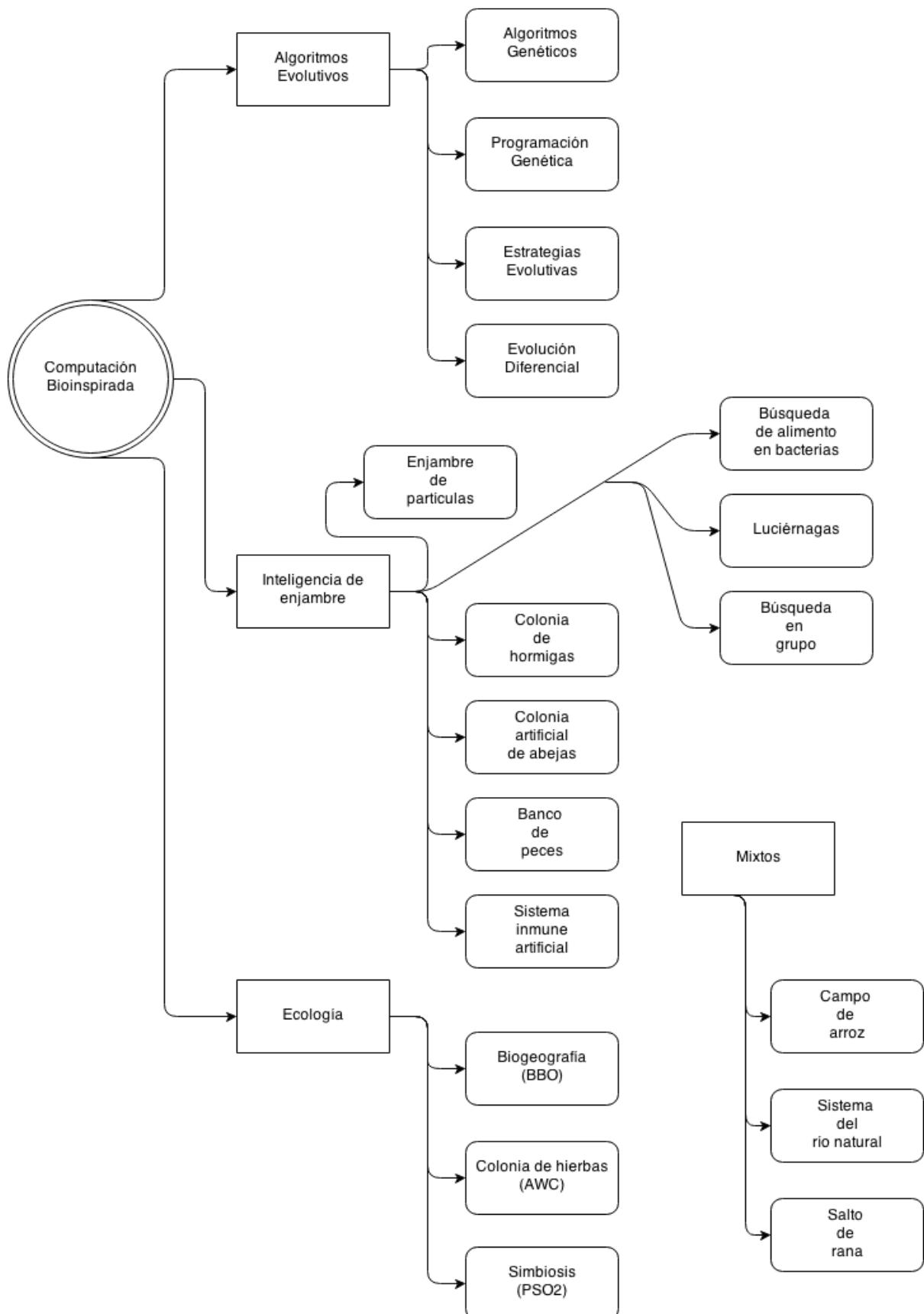
Debido a estas características muchos investigadores crearon nuevos métodos imitando la naturaleza, dando origen a los **algoritmos bioinspirados**. Al principio se usaron métodos específicos para problemas, pero más recientemente se han ido añadiendo métodos que funcionan en un amplio rango de problemas, pudiendo aplicarse de forma general.

En cuanto a sus procedimientos, estos algoritmos se centran en buscar la solución óptima a un problema, en el que todas sus posibles soluciones se pueden representar gráficamente como puntos en el espacio (la **nube de soluciones** del problema). Cada posición o punto en este

espacio es una solución. Los algoritmos de optimización cuentan con una función que se encarga de comprobar lo buena que es la solución, lo que se ha llamado **fitness** de la solución. Las soluciones óptimas, es decir las que tienen mayor fitness, se encuentran dispersas aleatoriamente en la nube de soluciones, con una disposición en la que conforme más se va acercando el logaritmo a cualquiera de las soluciones óptimas, el fitness va a ir aumentando progresivamente hasta llegar a esta. Cada uno de estas soluciones óptimas se conoce como **Óptimo local** (solución con más fitness en una región del espacio, por ejemplo pueden ser mínimos o máximos del problema), de entre los cuales uno de ellos tendrá el mayor fitness de todas las soluciones, siendo el **Óptimo global** (por ejemplo, el máximo o el mínimo del problema). Debido a la característica azarosa de estos algoritmos, por ser estocásticos, es posible que en el proceso el algoritmo pierda algunos óptimos locales, e incluso el óptimo global, lo que llevaría al algoritmo a quedarse atascado en un óptimo local, por lo que no encontraría la mejor solución al problema. Para evitar este fenómeno, muchos algoritmos cuentan con procedimientos para asegurar que se encuentran el mayor número de óptimos posibles, dotando al algoritmo de lo que se conoce como **habilidad de búsqueda global**.

El *objetivo* de este trabajo es revisar los principales, más usados y más eficaces algoritmos de computación bioinspirada, procediendo a describir en qué proceso, comportamiento o fenómeno biológico se basa cada uno de ellos. Se va a indicar además de qué forma es adaptada esta base biológica al algoritmo y se procederá a describir brevemente los pasos que se realizan en cada uno de ellos para llevar a cabo la optimización. También se mostrarán las ventajas propias de cada algoritmo así como sus principales usos y aplicaciones.

Se procederá utilizando el siguiente esquema:



Esquema: Tipos de bioalgoritmos

Como se observa en el gráfico, los métodos de optimización bioinspirados se dividen principalmente en tres categorías, clasificándolos según su base y los procedimientos que utilizan: si se basan es en procesos evolutivos son los Algoritmos Evolutivos, si es en la inteligencia colectiva que muestran poblaciones de individuos interactuando son Algoritmos de Inteligencia de Enjambre, y si es en las interacciones de las especies con otras especies y con su medio, son los Algoritmos basados en Ecología. Los Algoritmos Mixtos son aquellos que tienen características de varios grupos de los mencionados anteriormente.

Así pues, el trabajo se organiza del modo siguiente:

- **Sección II:** descripción de cada algoritmo que se encuentran divididos en categorías como se muestra en el gráfico.
- **Sección III:** comparación de los distintos algoritmos en una tabla, mostrando brevemente sus principales operadores, parámetros de control, aplicaciones y ventajas.
- **Sección IV:** Conclusiones del estado actual y futuro de la computación bioinspirada basadas en la información que se ha utilizado en este trabajo

II. Descripción de los algoritmos

II.1 Algoritmos evolutivos:

Este tipo algoritmos usados en búsqueda de soluciones no deterministas y en optimización, también conocidos como EA (Evolutionary Algorithm), son aquellos algoritmos dentro de la computación bioinspirada que se basan en las leyes evolución, tales como la selección, las mutaciones y la reproducción[2][3]. Los métodos más usados entre estos algoritmos son los Algoritmos Genéticos (GA), la Programación Genética (GP), la Evolución Diferencial (DE) y Estrategias Evolutivas (ES). Otro menos usado, como los Algoritmos Culturales.

Todos están basados en poblaciones de soluciones sometidas al criterio de la “supervivencia” de las mejores en un determinado “ambiente”, que es el problema a solucionar, seleccionándolas para la siguiente generación. Se parte de una población de soluciones posibles aleatorias, la cual va siendo seleccionada y cambiando hacia una solución mejor en cada generación.

-Algoritmo Genético:

(Genetic Algorithm, GA)

Están basados en el principio de la supervivencia del más apto, o fitness, de la Teoría de la Selección Natural de Darwin[2][3]. Las soluciones suelen tener forma de vector, y cada

solución se llama “**cromosoma**”. Los cromosomas son sometidos a procesos de **selección, reproducción (sobrecruzamiento) y mutación**. Para esto en cada generación se siguen unos pasos:

- 1- **Selección** de los mejores cromosomas, según lo buena que sea su solución al problema, a través de una función que mide el “fitness” de cada uno.
- 2- Los mejores cromosomas son los utilizados en el proceso de **reproducción o sobrecruzamiento** en el que se intercambian fragmentos de vectores entre cromosomas a partir de un punto determinado aleatoriamente en el cromosoma. También se hace **mutar** a los cromosomas cambiando valores aleatorios de forma aleatoria.

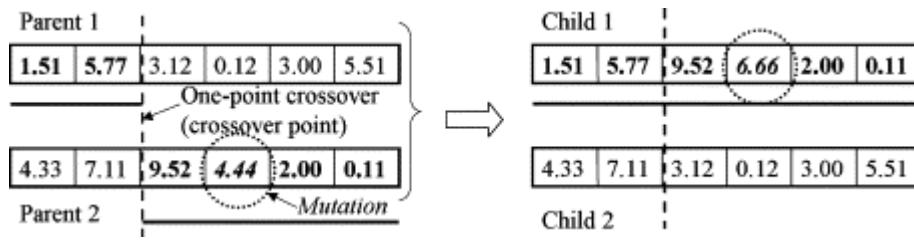


Ilustración 1: Sobrecruzamiento y mutación en los cromosomas. [2]

- 3-Se forma una nueva generación de cromosomas como resultado del paso anterior, que vuelve a ser evaluada para dar la siguiente generación.

Es el método más utilizado y que mejores soluciones da dentro de esta categoría, además de ser flexible en los tipos de fórmulas matemáticas que se pueden utilizar. Sobresalen en problemas de optimización, dando soluciones óptimas en un corto espacio de tiempo.

Son algoritmos que funcionan bien en problemas poco conocidos, difíciles, en los que fallan otros métodos. Además pueden ser usados para varios criterios y se pueden analizar muchas poblaciones de cromosomas grandes a la vez con poco coste computacional comparado con otros métodos.

Presenta algunos inconvenientes como que la solución final que consigue puede quedarse atascada en óptimos locales, que son difíciles de usar con datos dinámicos y que dependiendo del problema otros métodos más sencillos pueden dar mejores soluciones.

Entre sus aplicaciones industriales este algoritmo está siendo usado en campos como la ingeniería de control y procesamiento de señales, robótica, reconocimiento de patrones, optimización de diseños, reconocimiento de idiomas o sistemas de clasificación.

-Programación Genética:

(Genetic Programming, GP)

Es un método que está muy relacionado con los Algoritmos Genéticos, siguiendo los mismos principios, pero se diferencia en la forma de las soluciones. Lo hace por un proceso en el que obtiene variantes de una posible solución, que son evaluadas directamente, y las soluciones utilizadas pueden tener longitud y formas variables (al contrario que los AG, que son vectores de la misma longitud) pudiendo ser estas programas[2][3]. Por lo tanto es uno de los pocos métodos del que se pueden obtener directamente programas.

En la selección de programas se siguen unos pasos idénticos a los de los AG, utilizando un programa de fitness que mide cómo de bien resuelve el problema un programa, comparando la solución que da este con la óptima. Después los programas con mejor fitness son utilizados en el **sobrecruzamiento** y la **mutación** para dar la siguiente generación de programas a evaluar.

Este algoritmo tiene aplicaciones en campos como reconocimiento de patrones, robótica, clasificación. Un interesante ejemplo en el campo de la biología[4] ha sido una modificación del test de Fisher que se utiliza para el diagnóstico de cáncer de mama a través del reconocimiento de patrones. Esta modificación permite solventar las limitaciones de este test, como que no es necesario conocer la distribución probabilista de los datos para aplicarlo, minimizando la dispersión dentro de una clase y aumentándola entre las clases. Esto permite a la modificación aprender directamente de los datos, mejorando el test de Fisher al aumentar su funcionalidad, eficacia y robustez (se puede aplicar a muchos más tipos de datos).

-Evolución Diferencial:

(Differential Evolution, DE)

Es un algoritmo similar a GA, utiliza vectores como soluciones y los mismos procesos de selección; pero se diferencia de este en el proceso de mutación[2][3]. En este algoritmo la mutación es el resultado de combinaciones aritméticas entre los individuos de la población, mediadas por unos parámetros que se van adaptando al estado de la población. Estos parámetros comienzan favoreciendo la **mutación** para generar variabilidad, y conforme avanza la población se va disminuyendo la mutación para favorecer la formación de los individuos más óptimos. De esta forma se consigue evitar que la población caiga en óptimos locales.

Este algoritmo ofrece varias ventajas: es fácil de implementar, converge rápidamente, es robusto y tiende a llegar al óptimo global y no quedarse en locales. Sin embargo, es sensible al ruido y el ajuste de parámetros es manual, lo que puede hacer difícil encontrar los parámetros para un determinado problema.

-Estrategias Evolutivas:

(Evolution Strategies, ES)

Algoritmos que también parten de los principios de la selección natural, sin embargo están basados en los mecanismos de evolución a nivel de individuos, como el fenotipo o la heredabilidad[5], en vez de los mecanismos a nivel del genoma, como los Algoritmos genéticos.

Su característica más destacada que tienen mecanismos auto-adaptativos que permiten controlar la **tasa de mutación** para conseguir una optimización aún mayor de este método. Esto se consigue al añadir parámetros de estrategia a cada individuo, que también pueden mutar y heredarse, al igual que el fitness de cada uno. Estos parámetros modifican al de mutación para ese individuo, consiguiendo el valor óptimo de mutación para la resolución del problema. Así puede conseguirse que un “individuo” con poco fitness llegue a un parámetro de mutación alto, ya que necesita cambiar mucho para conseguir mejores resultados, y un “individuo” con mucho fitness mutará menos, ya que cambios excesivos podrían alejarlo de la solución.

Se utilizan diferentes parámetros de selección en las Estrategias Evolutivas, algunos de los cuales son:

(1-1)-ES: En cada generación, cada individuo de la generación parental es comparado con el individuo de la generación hija que se obtiene al aplicarle el parámetro de mutación. Para esto se evalúa la generación hija obteniendo su fitness y comparándolo con su parental, eliminando posteriormente el individuo con el fitness más bajo.

($\mu+\lambda$)-ES: En cada generación se seleccionan los mejores individuos para obtener la generación hija a través de mutación y recombinación. Después se suman los parentales y los hijos, y se analizan para obtener los mejores individuos de la siguiente generación.

(μ,λ)-ES: Se obtiene una generación hija a través de la mutación y recombinación de los mejores individuos de la generación parental, y luego se descarta la generación parental al completo.

Usados en optimización, se caracterizan por su flexibilidad: el poder utilizar distintos tipos de estrategias hace que se puedan adaptar mejor a un tipo de problema u otro.

II.2 Algoritmos de inteligencia de enjambre:

De la misma forma que los algoritmos evolutivos están basados en las leyes de la selección natural, los algoritmos de inteligencia de enjambre se basan, en sus comienzos, en el estudio del comportamiento de insectos sociales como las hormigas o las abejas, ya que tienen una alta capacidad de organización siguiendo normas muy simples[6]. Este comportamiento social se caracteriza por ser autónomo, auto-regulable, y funcionar de una forma descentralizada. Posteriormente su aplicación se ha extendido a otros tipos de agrupaciones animales tales como las bandadas de pájaros, la comunicación entre luciérnagas o incluso se han llegado a basar en el funcionamiento del sistema inmune.

Como ya hemos mencionado, estos algoritmos se basan en la inteligencia emergente de la interacción de agentes de comportamiento simple. Para conseguir esto, los algoritmos de inteligencia de enjambre siguen 5 principios básicos:

- 1) **Principio de proximidad:** La población debe ser capaz de hacer cálculos de espacio y tiempo sencillos.
- 2) **Principio de la calidad:** La población tiene que ser capaz de reconocer cambios de la calidad en el “ambiente” (es decir, en la nube de soluciones en las que se está ejecutando la población)
- 3) **Principio de la respuesta diversa:** La población no puede ejecutar su actividad en zonas de soluciones excesivamente pequeñas.
- 4) **Principio de estabilidad:** La población no debe cambiar su comportamiento cada vez que cambie el “ambiente”
- 5) **Principio de la adaptabilidad:** La población debe cambiar su comportamiento cuando merezca la pena el esfuerzo computacional requerido.

Estos dos últimos principios se utilizan para regular el comportamiento de las poblaciones.

-Optimización por enjambre de partículas:

(Particle Swarm Optimization, PSO)

Este método se basa en utilizar las bases de la inteligencia artificial para emular poblaciones con comportamientos sociales simples como los de las bandadas de aves o los bancos de peces, en las que muchos individuos tienen en cuenta la velocidad, dirección y la posición de

los individuos que los rodean para moverse como un conjunto[7].

Para imitar este comportamiento, se establece un enjambre de “**partículas**” (que son agentes que no tienen masa o volumen, a los que se les somete a velocidad y aceleración distintas para buscar una solución) que serían puntos en la nube de soluciones, capaces de hacer cálculos sencillos. Cada partícula tiene dos parámetros, la **posición** que es la solución de la nube en la que se encuentra actualmente, y la **velocidad** a la que se mueve por los ejes de la la nube de soluciones. (determinando así también la dirección de la partícula). Además cada partícula es capaz de “recordar” la posición con mejor fitness en la que se ha encontrado (pbest) y la posición con mejor fitness de todas las partículas vecinas (gbest). En cada ciclo la nueva posición de la partícula viene determinada por su posición anterior y por la velocidad, que depende de su velocidad anterior pero que también está influenciada con una intensidad aleatoria por pbest y gbest. De esta forma se guía a la partícula hacia una zona con mejor soluciones.

En su ejecución se siguen los siguientes pasos:

- 1) Iniciación, se coloca cada partícula del enjambre en una posición aleatoria en la nube de soluciones.
- 2) Evaluación del fitness de las partículas.
- 3) Comparación del valor del fitness de cada partícula con su pbest correspondiente. En el caso de ser mayor, se sustituye el pbest por la nueva posición.
- 4) Identificación de la partícula con mejor fitness sustituyendo gbest por el valor de su posición.
- 5) Modificación de los parámetros de cada partícula y cálculo de su nueva posición.
- 6) Repetición de los pasos hasta que se llegue al número de ciclos indicado o a un fitness aceptable.

Este método tiene varias ventajas:

Eficiente: se implementa muy fácilmente al ser muy sencillo, y además da unos buenos resultados.

Capacidad de memoria superior a los Algoritmos Evolutivos.

Aumento de variabilidad: ya que cada partícula usa su mejor resultado y el mejor resultado de todas para alcanzar mejores valores, en vez de eliminar los peores valores como ocurría en los Algoritmos Evolutivos. Así se consigue aumentar la variabilidad de la población.

Principalmente utilizado para optimización de funciones continuas no lineares de forma estocástica.

-Optimización de la colonia de hormigas:

(Ant Colony Optimization, ACO)

Es un método inspirado por el comportamiento de búsqueda de comida que realizan las colonias de algunas especies de hormigas, por una forma de comunicación conocida como estigmergia (forma de comunicación no simbólica, local, que se produce al modificar el ambiente.)[8] Esto se debe a que, mientras las hormigas van desde la colonia a las fuentes de alimento y viceversa, van dejando un rastro de feromonas. Ya que las hormigas que pasan por caminos más cortos tardarán menos tiempo en moverse entre el recurso y la colonia, este camino tendrá un rastro de feromonas mayor al pasar más veces las hormigas por él. De esta forma las hormigas consiguen ir siempre por el camino más corto hasta un recurso, ya que las otras hormigas de la colonia siguen los que presentan un rastro con feromonas más concentradas.

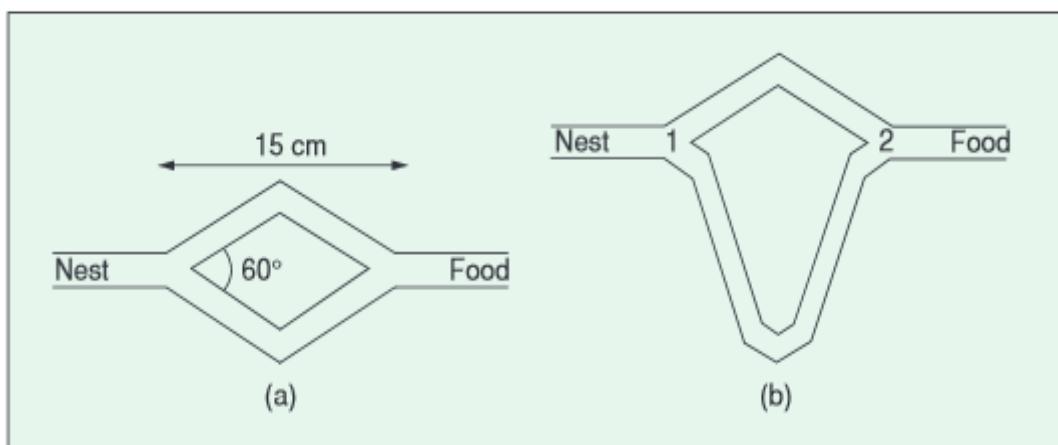


Ilustración 2: Selección de camino en hormigas. a) Se elige un camino aleatorio. b) Se elige el camino superior, ya que las hormigas tardan menos en ir y venir, dejando más feromonas. [8]

El algoritmo basado en Colonia de hormigas utiliza unas **hormigas artificiales** que van de unos puntos a otros del problema dejando un rastro de “feromonas”. Tiene tres procedimientos principales:

AntSolutionsConstruct: construir las distintas soluciones al problema, en las que las “hormigas” se van moviendo de unos puntos a otros de acuerdo a una regla de transición, dando así diferentes soluciones

PheromoneUpdate: Actualizar los rastros de “feromonas”. Esto puede hacerse una vez que las hormigas artificiales acaben de pasar por los puntos, o cada vez que pasan por un punto. También se incluye una función de evaporación que se encarga de ir reduciendo los rastros, de forma que los que menos concentración tienen van

desapareciendo gradualmente.

DaemonAction: reforzar el rastro de “feromonas” de los mejores caminos generados. Este procedimiento no se da en la naturaleza, pero sirve para mejorar la eficacia del algoritmo.

Ya que este algoritmo se utiliza para buscar el camino mas corto entre un número variable de localizaciones o puntos, tiene aplicación en campos como la industria, telecomunicaciones, transportes, etc; además de en optimización. Es un algoritmo meta-heurístico muy eficaz.

Este algoritmo ya está siendo utilizado por algunas industrias para aumentar su eficacia. Por ejemplo, en una fábrica de piezas de aluminio llamada Alcan [20], la cual responde a pedidos de clientes en los que puede variar el número de piezas, su tamaño, acabado, etc; aplican este algoritmo a conjuntos de pedidos para que les indique el orden para cumplir los objetivos de todos los pedidos, y así reducir el tiempo de producción al tener que cambiar menos la configuración de las máquinas. El algoritmo ha demostrado proveer de buenas soluciones en un corto periodo de tiempo y con bajo esfuerzo computacional.

Otro ejemplo es en una instalación de mantenimiento de trenes del Sistema Ferroviario Francés[21], para resolver un problema de distribución de las instalaciones. Es decir, encontrar una buena disposición de las máquinas, equipamiento y otros recursos para reducir el tiempo de producción, optimizando los flujos de producción y reduciendo así su coste. En este caso, al utilizar el algoritmo mostró una nueva disposición para las instalaciones que reducía los costes casi un 20% respecto a la disposición original, lo que permitirá a estas instalaciones reducir sus gastos y aumentar su eficacia.

-Algoritmo de la Colonia Artificial de Abejas

(Artificial Bee Colony, ABC)

Otro de los algoritmos basados en comportamientos animales. La base de este algoritmo es la inteligencia colectiva resultante de los comportamientos sencillos que desarrollan las abejas. Aunque hay otros tipos de algoritmos basados en abejas (por ejemplo en su comportamiento de selección de pareja), el más conocido y representativo es el de la Colonia Artificial de Abejas. Este se fundamenta en el comportamiento de búsqueda y explotación de recursos de las abejas[9].

Utiliza 3 tipos de **abejas artificiales**: abejas **empleadas**, abejas **observadoras** y abejas **exploradoras**. Las primeras van a fuentes de alimento que previamente han visitado ellas mismas, las segundas esperan en el “área de danza” para decidir que fuente de alimento

visitar, y las tercera viajan aleatoriamente buscando nuevas fuentes de alimento. La posición de cada fuente de alimento representa una solución de la nube de soluciones, y la cantidad de néctar su fitness.

En cada ciclo se producen los siguientes pasos:

- 1- Cada abeja empleada produce una modificación de su fuente de alimento, para comprobar si en las zonas vecinas hay más cantidad de néctar. Si la hay, la abeja escoge esta nueva posición y olvida la anterior, si no sigue con su posición original.
- 2- Posteriormente se dirigen a la zona de danza, donde comparten la información del néctar de su zona con las abejas observadoras. Estas reúnen la información de las abejas empleadas y acompañan a una de estas a su fuente de alimento, siendo más probable que ocurra esto cuanto más néctar tenga esa fuente. Una vez allí hacen un proceso similar al de la abeja empleada, produciendo una modificación de la fuente y sustituyendo la solución anterior que tienen guardada en memoria si tiene más néctar, en caso contrario no se modifica.
- 3- En el último paso las abejas empleadas cuya posición esté agotada (no ha mejorado ni ha sido escogida en un número variable de ciclos) se convierten en exploradoras. Por último se envían las abejas exploradoras a nuevas posiciones aleatorias, y volverán a ser abejas empleadas.

La población de la colonia consiste en la mitad de abejas empleadas y la mitad de observadoras, habiendo sólo fuente de alimento por cada empleada y seleccionándose en cada ciclo las abejas exploradoras.

El proceso se para cuando se cumplen determinadas condiciones de fitness o se llega a un número de ciclos.

Este algoritmo se caracteriza por su alto rendimiento comparado con otros algoritmos así como capacidad para salir de óptimos locales y encontrar el óptimo global. Sin embargo, los parámetros de control que se le impongan al inicio pueden afectar su rendimiento y su velocidad.

Se puede emplear eficazmente en problemas de optimización complejos, como multimodales y multivariados. También ha demostrado su utilidad en aplicaciones industriales como problemas de ordenamiento de trabajos, es decir el orden en el que se deben encontrar los diferentes procesos de una cadena industrial para mejorar su eficacia y reducir el tiempo de producción.

-Algoritmo del banco de peces

(Fish Swarm Algorithm, FSA)

Es un algoritmo de optimización propuesto recientemente, en este caso basado en el comportamiento de las poblaciones de los bancos de peces a la hora de la búsqueda de alimento[10][11]. Los peces del banco pueden optar por dos estrategias, buscar alimento por sí mismos o seguir a otros individuos a las zonas con más peces, que suelen ser las que más alimento tienen.

Para conseguir esto, se programan **peces artificiales** que se mueven por una nube de soluciones en busca de las zonas con mejor calidad nutricional (mejores soluciones). Se establecen 3 tipos de comportamientos:

1-Búsqueda: Cada pez artificial se desplaza por la nube de soluciones en búsqueda de zonas con alimento.

2-Agrupamiento: A parte de obtener información de la zona en la que está cada pez artificial, estos también obtienen información de los peces vecinos sobre la calidad de su zona, lo que hace que los peces se desplacen a las mejores zonas de su rango si no están muy saturadas de otros peces.

3-Seguimiento: Cuando un pez encuentra alimento, los peces de alrededor tienden a seguirlo, más peces cuanto mayor sea la calidad de la nueva zona.

Por lo tanto el comportamiento de cada pez no depende sólo de la zona en la que se encuentre, si no también de los hallazgos de los otros peces en su rango.

Este algoritmo tiene muchas ventajas: es muy robusto, habilidad de búsqueda global, y es muy tolerante con los valores de sus parámetros y los valores iniciales de la población. Sin embargo la búsqueda global puede ser lenta ya que muchos peces artificiales tienden a buscar el óptimo de una zona y no el global. A causa de esto ha salido una versión mejorada del algoritmo en la que se comparte información de toda la población, y no sólo de los peces vecinos, para paliar esta desventaja.[10]

-Algoritmo de optimización basado en búsqueda de alimento de bacterias:

(Bacterial Foraging Optimization Algorithm, BFA)

Es un algoritmo propuesto recientemente basado en el comportamiento de búsqueda de alimento de la bacteria *E.coli* [12]. Estas bacterias realizan este comportamiento en conjunto

más que individualmente para maximizar la cantidad de alimento obtenido en el menor tiempo posible, comunicándose entre ellas mientras lo realizan. Cada bacteria utiliza un proceso llamado **quimiostasis** (movimientos hacia o en contra una señal química percibida por la bacteria) en la búsqueda de alimento, que realizan a través de pequeños desplazamientos.

Para emular este comportamiento, el algoritmo utiliza 4 mecanismos:

1-Quimiostasis: Imita el comportamiento de movimiento la bacteria. Para desplazarse en un gradiente de nutrientes (soluciones) a través de la quimiostasis la bacteria puede hacer dos movimientos, avanzar o girar. Las bacterias artificiales pueden variar entre estos dos modos de movimiento en cualquier momento.

2-Formación de enjambres: Imita un comportamiento de comunicación celular observado en algunas bacterias con movilidad, en el cual al detectar altas concentraciones de nutrientes liberan una señal celular para atraer a más congéneres, resultando en una formación de anillo con alta densidad bacteriana, que se mueve como un conjunto hacia los nutrientes.

3-Reproducción: Las bacterias que menos alimento encuentran acaban muriendo, y en cambio las que más alimento encuentran se dividen dando 2 bacterias en la misma posición, manteniendo el tamaño de la población constante y emplazando más bacterias en las zonas con más nutrientes.

4-Eliminación y dispersión: añade un factor aleatorio al imitar las condiciones ambientales de las bacterias, en las que algún grupo de bacterias puede morir o ser dispersadas hasta otras localizaciones. Para esto algunas bacterias pueden ser eliminadas aleatoriamente con una baja posibilidad y se emplazan nuevas bacterias al azar en la nube de soluciones. Esto asegura la diversidad y evita que la función caiga en óptimos locales.

Este algoritmo está haciendo popular debido a que ha demostrado ser más eficaz que otros algoritmos de inteligencia de enjambre y genéticos. Además tiene aplicaciones industriales en campos como la ingeniería electrónica, reconocimiento de patrones y problemas de ordenamiento de trabajos.

-Algoritmos del sistema inmune artificial:

(Artificial Immune System Algorithm, AIS)

Son algoritmos basados en las distintas estrategias que utiliza el sistema inmune humano en el

reconocimiento de antígenos y la defensa del organismo, así como en la formación de sus componentes [13][14]. Es un sistema adaptativo muy evolucionado y complejo, descentralizado y está basado en la interacción entre sus distintos componentes.

El más conocido y más usado dentro de esta categoría se basa en la selección y mutación de clones de linfocitos B cuando reconocen un antígeno. Para emular este comportamiento se realizan los siguientes pasos:

1-Iniciación: Se establecen los anticuerpos (soluciones), siendo los antígenos el valor de la función a ser optimizada.

2-Clonación: se comprueba el fitness de cada anticuerpo según su afinidad al antígeno. Los anticuerpos se cloran un número de veces que depende de su afinidad, cuanto más alta más clones se producen.

3-Hipermutación: Los clones se someten a un proceso de mutación inversamente proporcional a su afinidad, para conseguir que los clones menos afines muten más y al contrario. Posteriormente cada clon mutado se compara con su original para ver si su fitness es superior y los de mayor fitness se usan en el siguiente ciclo.

Además de este método también hay otros como la Selección Negativa, basado en el proceso de selección negativa que sufren los linfocitos T en el timo para evitar que reconozcan antígenos propios, eliminando a los linfocitos que lo hacen.

Este algoritmo tiene varias ventajas: capacidad de reconocimiento, aprendizaje, memoria, diversidad y robustez. En cuanto a sus aplicaciones, a parte de su uso en optimización de funciones, este algoritmo funciona muy bien en el reconocimiento de patrones (se aplican los mismos métodos, pero comparando patrones en vez de soluciones). Esta última característica lo hace especialmente útil en campos como seguridad informática, detección de errores, detección de datos nuevos en análisis, clasificación de datos, etc.

-Algoritmo de las luciérnagas:

(Firefly Algorithm, FA)

Es un algoritmo basado en poblaciones, que se inspira en el comportamiento de comunicación por ráfagas de luces de las luciérnagas [15]. Estas utilizan la bioluminiscencia generada en un órgano llamado linterna para lanzar distintos patrones de ráfagas de luz, que les sirven para transmitir información como la especie o el sexo del individuo, advertir a depredadores (muchas especies son venenosas), o incluso para la caza de otras especies de luciérnagas.

El algoritmo se utiliza para buscar soluciones aleatoriamente en una nube de soluciones, pero con comunicación entre los agentes de búsqueda (**luciérnagas**). Esta comunicación se basa en tres premisas:

- A- Todas las luciérnagas son **unisex**, por lo que son atraídas por todas las luces.
- B- La atracción que sienten las luciérnagas por una luz es directamente proporcional a la **intensidad** de la luz emitida.
- C- La intensidad de la luz que emite una luciérnaga depende del **fitness** de su posición (solución). Cuanto mayor sea el fitness, mayor es la intensidad. Además, la luz emitida va perdiendo intensidad según la distancia.

Por lo tanto, al empezar a buscar soluciones con las luciérnagas distribuidas aleatoriamente, cada una comunicará a los otros agentes cercanos el fitness de su posición, atractando a más luciérnagas cuanto mayor sea el fitness de la posición. Así se consigue dirigir la búsqueda de soluciones hacia las mejores posiciones y haciéndola mucho más eficaz que una búsqueda aleatoria normal.

Sus principales usos están en los campos de la optimización de forma heurística, la clasificación y en algunos campos de la ingeniería como diseño de antenas, procesamiento de imágenes o redes inalámbricas.

-Optimizador de búsqueda de grupo

(Group Search Optimizer, GSO)

Es un método de optimización basado en poblaciones e inspirado en el comportamiento de búsqueda (de alimento, pareja, zonas para anidar, etc.) de los animales[16]. Particularmente está basado en la teoría del **Productor-Parásito** (Producer-Scrounger), que es propio de las especies que forman grupos (en este caso, el término parásito se refiere al parasitismo intraespecífico). El grupo permite que la búsqueda tenga muchas más posibilidades de éxito al haber más individuos que busquen el recurso, pero esto lleva a la adopción de dos estrategias en el grupo: la de productor, que busca activamente recursos, y la de parásito, que explota los recursos descubiertos por otros.

Para imitar este comportamiento se hacen poblaciones artificiales llamadas **grupos**, en las que cada individuo se llama **miembro**. Se establecen tres tipos de miembros, los **buscadores**, los **parásitos** y los **miembros dispersos** que se mueven al azar por la nube de soluciones. Se asume que cada miembro sólo puede ser de un tipo a la vez, pero pueden cambiar de tipo tantas veces como se desee.

Para simplificar el modelo se selecciona un único productor a la vez, que en cada ciclo es el miembro que esté en la posición con el mejor fitness. El resto son parásitos, que se dirigen hacia el productor, o dispersos, que se mueven al azar. El productor elegido se para en su posición y escanea el área circundante para localizar una mejor posición que la actual. Si la localiza, “salta” a esa nueva posición, si no permanece en la posición en la que está. El comportamiento de escaneo se hace de forma “visual”, para lo cual el productor tiene un campo visual en forma de cono (de ángulos, altura y profundidad determinados) cuya dirección depende de un ángulo generado al azar. La dirección del cono está regida por un ángulo elegido al azar. En el caso de no encontrar una posición mejor que la actual, se genera otro ángulo al azar.

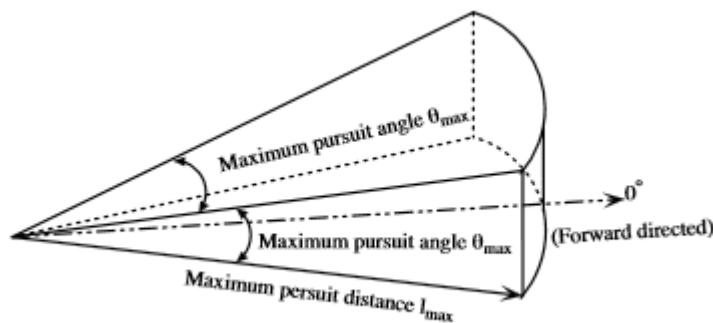


Ilustración 3: Ángulo de visión de los miembros [16]

Si cualquier miembro parásito o disperso llega a una posición mejor que la del productor, en la siguiente ronda se convertirá en el nuevo productor.

Es un algoritmo usado para la optimización, se caracteriza por su alta precisión y velocidad de convergencia.

II.3 Ecología:

Los algoritmos de esta categoría están basados en los ecosistemas, que también son buenos ejemplos para resolver problemas. En este ámbito se incluyen los algoritmos que se basan en las interacciones de los animales con el ecosistema en el que habitan. Estas interacciones pueden ser entre el animal y su medio (suelo, agua, aire, etc.), relaciones con la propia especie (animales sociales, competencia intraespecífica) o con otras especies del ecosistema, tanto antagonistas (depredación, competencia, parasitismo...) como de cooperación (mutualismo). También se incluye la biogeografía.

-PS2O:

Es un algoritmo que se basa en la coevolución de varias especies[17], las cuales evolucionan juntas obteniendo beneficios una de la otra, como alimentos, protección, etc. Este hecho puede tratarse como un proceso de optimización. Este método es una modificación del algoritmo PSO (Optimización del enjambre de partículas) en el que en vez de utilizar una sola población de individuos, se utilizan varias poblaciones (**especies**) que interaccionan entre sí mismas y las unas con las otras para encontrar una solución mejor (**coevolución**).

Con este propósito se establecen varias poblaciones con un número de individuos cada una, y todas ellas reciben la misma presión evolutiva (buscan la solución al mismo problema). Cada individuo tiene un valor de fitness y evoluciona según recibe información de 3 fuentes distintas; la encontrada por el propio individuo, la que obtiene de los otros miembros de su especie, y la que obtiene de individuos de otras especies de la comunidad; a la vez que comparte la información que él encuentra. El proceso de selección de individuos para determinar cuáles comparte información y cuáles tienen que cambiar su posición son los mismos que los del algoritmo PSO.

Para generar variabilidad, después de cierto número de generaciones se eligen la mitad de las especies y se eliminan, a la vez que se establece un mismo número de especies aleatoriamente.

Al añadir la interacción entre especies al algoritmo PSO, se consigue una mayor velocidad de convergencia y también se consigue evitar el problema de que la población se quede en óptimos locales y no el general, como ocurre a veces con el PSO.

-Optimización de las malas hierbas invasoras:

(Invasive Weed Colony Optimization, IWO)

Es un algoritmo de optimización basado en el comportamiento de las plantas herbáceas que invaden cultivos[18]. Esta idea surgió al darse cuenta de que, a pesar de todo el esfuerzo en eliminar este tipo de plantas a lo largo de la historia, no se ha conseguido acabar con ninguna especie de estas plantas y además se han ido haciendo cada vez más resistentes (a los herbicidas por ejemplo). Es decir, se caracterizan por su alta robustez para aguantar el estrés al que las sometemos, así como una alta flexibilidad para adaptarse a este. Son pues un buen ejemplo para problemas de optimización.

Para imitar el comportamiento de estas plantas se ejecutan los siguientes pasos:

1-Establecimiento de la población inicial: Se colocan aleatoriamente individuos en la nube de soluciones.

2-Reproducción: Se mide el fitness de cada individuo de la población. Posteriormente cada individuo produce un número de semillas directamente proporcional a su fitness.

3-Dispersión: Cada planta dispersa sus semillas a su alrededor, en un área que va disminuyendo conforme van pasando más generaciones. Esto asegura variabilidad al comienzo del proceso, y una búsqueda más precisa de soluciones en las mejores zonas al final de este.

4-Exclusión competitiva: Cuando se alcanza el máximo número de individuos, estos producen semillas y dispersan semillas normalmente, pero a cada semilla se le mide el valor de fitness y se eliminan las semillas con menor fitness. Se mide el fitness cuando las semillas son dispersadas para permitir que las plantas con fitness bajo también se reproduzcan, ya que podrían llegar a nuevas zonas en el que este fuera más alto.

Se caracteriza por ser un algoritmo rápido, capaz de resolver problemas de optimización complejos (sobre todo de funciones objetivas), no tiende a caer en óptimos locales y es eficaz encontrando buenas soluciones.

-Optimización basada en Biogeografía:

(Biogeography-Based Optimization, BBO)

Es un algoritmo de optimización basado en la biogeografía[19], es decir, en el estudio de la distribución de las especies en el tiempo y el espacio, así como las causas que han llevado a esta situación. Este algoritmo se centra en la emigración y la inmigración, permitiendo el intercambio de información entre posibles soluciones.

En este algoritmo a cada posible solución (en forma de vector) se la conoce como “**hábitat**”, teniendo cada hábitat sus propias características (componentes del vector) llamadas **variables de índice de idoneidad** (suitability index variables (SIV)), cuyos valores determinan la capacidad del hábitat de albergar especies o **índice de idoneidad del hábitat** (habitat suitability index (HSI)) siendo mejor hábitat cuando mayor sea el HSI (o fitness de la solución). Los hábitats comparten información a través de la “**migración**” de características (SIV) del hábitat. Cada hábitat tiene unos índices de inmigración y emigración que dependen del HSI del hábitat, siendo la inmigración menor cuanto mayor sea HSI, y la emigración mayor cuanto mayor sea HSI. Estos índices aseguran que los mejores hábitats compartan más información y que los peores reciban más de estos.

Además se incorpora un componente de aleatoriedad y diversidad en forma de mutaciones,

cada hábitat tiene una probabilidad de mutar a otro hábitat, que es mayor cuanto menor sea su HSI.

Es un método que se caracteriza por no descartar ninguna solución, sólo las modifica, y en que los índices de migración dependen del fitness. Por otra parte, no destaca en velocidad o bajo consumo de recursos, está a la par con otros métodos.

II.4 Mixtos:

Estos algoritmos se encuentran en una categoría aparte ya que utilizan elementos de varios de los tipos mencionados anteriormente, pudiendo ser su base, los elementos que utilizan, el tipo de algoritmo, etc.

-Algoritmo del campo de arroz:

(Paddy Field Algorithm, PFA)

Está basado en el crecimiento de una población de plantas en un campo de arroz, aunque también usa los principios de los algoritmos evolutivos[22].

En vez de basarse en el sobrecruzamiento y mutación, depende de unos parámetros llamados **polinización y producción de semillas**. El primero depende de la **densidad** de la población en una zona (cuanto más densamente poblada esté una zona, más soluciones se obtendrán de esta) y la otra depende del fitness de la planta. Estas semillas se dispersan en la nube de soluciones en busca de mejores zonas. De forma similar a los algoritmos evolutivos, se eliminan los individuos con peor fitness en cada ciclo.

Se caracteriza por su bajo coste computacional.

-Algoritmo de las gotas de agua inteligentes:

(Intelligent Water Drops Algorithm, IWD)

Es un algoritmo basado en ecología, en concreto en el comportamiento de los ríos a la hora de elegir el mejor camino para llegar a su destino a lo largo del tiempo (las zonas más rápidas del río tienen más erosión, haciendo que estas zonas se amplíen y captén más agua; mientras que las más lentas tienden a acumular sedimentos, estrechándolas.) Sin embargo este algoritmo considera las gotas como individuos, por lo que está basado en poblaciones[23].

Para imitar el comportamiento de las gotas en la naturaleza, cada **gota** (individuo) tiene dos parámetros: la **cantidad de tierra** (soil) que contiene y la **velocidad** actual. Las gotas se van

moviendo en pequeños pasos, buscando el mejor camino en cada uno. En cada paso, la velocidad de la gota es inversamente proporcional a la cantidad de tierra del camino. Además cuanto más rápido vaya la gota, más tierra se le añade a esta. Por último, las gotas tienen más probabilidad de escoger un camino cuanta menos tierra tenga este. Con este método se van seleccionando los mejores caminos sin tener conocimiento de estos.

Este algoritmo recientemente propuesto ha mostrado su gran eficacia tanto en problemas de búsqueda de camino más corto como de optimización.

-Algoritmo de salto de las ranas barajadas:

(Shuffled Frog Leaping Algorithm, SFLA)

Combina las ventajas de algoritmos evolutivos con el comportamiento social de los algoritmos de enjambre, como el PSO. Está basado en el comportamiento e intercambio de información que se produce mientras las ranas buscan alimento situadas en piedras en un estanque[24].

En el algoritmo, cada individuo de la población es una **rana** (son idénticas a los cromosomas de los algoritmos evolutivos, cada una con su propio fitness), y estas son divididas en subpoblaciones llamadas **memeplex**. Cada memeplex evoluciona de manera independiente y cada cierto número de ciclos se intercambian ranas aleatorias entre los distintos memeplex, para ayudar a alcanzar el óptimo global.

Se caracteriza por su sencillez, eficacia y rapidez.

III. Tabla comparativa:

En esta sección se mostrará una tabla de los algoritmos vistos, para facilitar su comparación centrándose en sus operadores, parámetros de control, ventajas, y algunas áreas de aplicación.

Nombre	Operadores	P. de control	Ventajas	Aplicaciones
GA	Sobrecruzamiento, mutación, selección.	Tamaño de la población, número máximo de generaciones, probabilidad de sobrecruzamiento, probabilidad de mutación, tamaño del cromosoma.	Flexible, puede ser utilizado en problemas difíciles y poco conocidos, bajo coste computacional.	Ingeniería de control y procesamiento de señales, robótica, reconocimiento de patrones, optimización de diseños, reconocimiento de idiomas o sistemas de clasificación.

GP	Sobrecruzamiento, mutación, selección.	Tamaño de la población, número máximo de generaciones, probabilidad de sobrecruzamiento, probabilidad de mutación.	Permite crear y optimizar programas enteros, aplicación con tamaño de solución variable.	Reconocimiento de patrones, clasificación, detección de cáncer.
DE	Mutación, selección, sobrecruzamiento.	Tamaño de la población, probabilidad de sobrecruzamiento, dimensión del problema, factores de escala	Fácil de implementar, converge rápidamente, robusto, capacidad de búsqueda global	Filtros digitales
ES	Mutación, selección, recombinación discreta	Tamaño de la población, número máximo de generaciones, probabilidad de sobrecruzamiento, probabilidad de mutación.	Flexibilidad, adaptables a distintos tipos de problemas	Estimación de parámetros, procesamiento de imágenes.
PSO	Inicializador, actualizador y evaluador.	Número de partículas, dimensión de partículas, rango de las partículas, velocidad máxima, factores de aprendizaje, número máximo de interacciones.	Eficaz, fácil de implementar, capacidad de memoria, alta variabilidad.	Optimización de funciones continuas no lineares, sistemas de energía eléctrica, procesamiento de imágenes...
ACO	Actualizador de feromonas, medidor de feromonas, evaporación de feromonas	Número de hormigas, tasa de evaporación de feromona, número de iteraciones, refuerzo de feromonas	Muy eficaz	Optimización, industria, telecomunicaciones, transportes...
ABC	Reproducción, reemplazamiento de abejas, selección.	Número de recursos de alimento (población de abejas observadoras), número máximo de ciclos	Alto rendimiento, capacidad de búsqueda global.	Problemas de optimización multivariadas y multimodales, ordenamiento de trabajos

FSA	Agrupamiento, seguimiento, búsqueda.	Distancia visual, máxima distancia de paso, factor de multitud.	Robusto, capacidad de búsqueda global, tolerante con sus parámetros.	Optimización, estimación de parámetros.
BFA	Reproducción, quimiostasis, dispersión, eliminación, agrupación.	Amplitud del espacio de búsqueda, número de bacterias, número de pasos, número de eventos de eliminación y dispersión, probabilidad de eliminación, probabilidad de reproducción, localización de la bacteria...	Muy eficaz	Ingeniería electrónica, reconocimiento de patrones, ordenamiento de trabajos.
AIS	Operadores de inmunidad (clonación, hipermutación, selección)	Tamaño de población, número de anticuerpos seleccionados por hipermutación, número de anticuerpos reemplazados.	Especialmente efectivo en reconocimiento de patrones.	Optimización, seguridad informática, detección de errores, detección de datos nuevos en análisis, clasificación de datos.
FA	Luminiscencia	Número de luciérnagas, campo de visión.	Mejora de búsqueda aleatoria, eficaz.	Optimización, procesamiento de imágenes, redes inalámbricas.
GSO	Índice de parasitismo, de dispersión y de producción.	Tamaño de la población, número de miembros dispersos, tamaño de ángulos máximo.	Alta precisión y velocidad de convergencia.	Optimización, problemas de flujo de corriente, diseño o monitorización de estado de desgaste.
PSO	Inicializador, actualizador , extinción y evaluador.	Número de partículas, dimensión de partículas, rango de las partículas, velocidad máxima, factores de aprendizaje, número máximo de interacciones.	Eficaz, fácil de implementar, capacidad de memoria, alta variabilidad, velocidad alta y capacidad de búsqueda global.	Optimización, construcción de sistemas de servicios cooperativos.

IWO	Inicializador, evaluador, reproducción, dispersión, selección.	Tamaño máximo de población, número de semillas, índice de dispersión.	Eficaz, rápido, capacidad de búsqueda global, aplicable a problemas complejos.	Optimización de funciones objetivas.
BBO	Migración (emigración e inmigración), mutación.	Número de hábitats, máximo índice de migración, índice de mutación.	A la par que otros métodos en velocidad y bajo consumo de recursos.	Optimización, procesamiento de imágenes.
PFA	Dispersión, polinización	Tamaño de la población, número máximo de semillas, límites del espacio.	Bajo coste computacional.	Optimización de funciones continuas.
IWD	Reproducción, dispersión, selección.	Tamaño de población, desviación estándar, índice de modulación.	Muy eficaz	Optimización, búsqueda de camino más corto.
SFLA	Reemplazamiento, índice de barajado	Tamaño de la población, número de memplexes, número de iteraciones antes del barajado.	Sencillez, eficaz, rápida convergencia.	Optimización.

IV. Conclusión:

Los algoritmos bioinspirados supusieron pasar de intentar resolver los problemas de optimización por medios tradicionales, que requerían más comprensión del mismo y no eran aplicables a otros más complejos, a copiar los modelos de optimización de algo que lleva mucho más tiempo resolviendo este tipo de cuestiones sin conocimiento previo y de forma eficaz: la naturaleza. Desde sus comienzos ya demostraron su potencialidad para resolver este tipo de problemas, y con el paso del tiempo han ido apareciendo más modelos que se pueden aplicar de forma más fácil y en más campos. De su uso casi exclusivo en ingeniería ha saltado a campos como la industria, la seguridad informática, la investigación, el diagnóstico, el diseño, la robótica.... El aumento de su sencillez y la reducción del esfuerzo computacional que requieren ha permitido que estos algoritmos sean utilizados por empresas, ofreciendo soluciones a problemas a los que se enfrentan, y aumentando así sus ganancias. Por lo tanto

no es extraño pensar que en un futuro próximo estos algoritmos puedan ser utilizados de forma cotidiana por las compañías para averiguar la mejor forma de llevar ciertas partes del negocio (la distribución, el transporte, la disposición de la maquinaria, etc) y así aumentar las beneficios con la misma inversión.

Al ser un campo de investigación relativamente joven aún queda mucho espacio para el desarrollo y la aparición de nuevos algoritmos bioinspirados. Estos permitirán no sólo el desarrollo de las ciencias de la computación si no también la resolución de problemas a los que nos enfrentamos en la actualidad, tanto en investigación, como en la industria e incluso en asuntos que nos afectan en el día a día, como los transportes públicos, la sanidad o la optimización de ciertas áreas de negocios.

Crítica a la bibliografía:

La bibliografía es fácil de localizar, la mayoría de los artículos proveen de toda la información necesaria de interés para el trabajo (descripción general, método, base biológica, usos y ventajas). Muchas de ellas tienen gran cantidad de contenido matemático e informático que era excesivo para el objetivo de este trabajo, por lo que no se ha tenido en cuenta. Los ejemplos específicos se han tenido que buscar aparte, siendo algo más difíciles de localizar, debido al número limitado de ejemplos de aplicación de estos algoritmos a nivel industrial u otros usos que no sean la optimización computacional.

V. Bibliografía:

Método de búsqueda bibliográfico:

Para la localización de la bibliografía mostrada en este trabajo, se buscó específicamente cada método en los buscadores de Google Scholar y The Web of Science, seleccionando los artículos que mejor describiesen la metodología del algoritmo.

[1]S. K. Pal, S. Bandyopadhyay and S. S. Ray
“Evolutionary computation in bioinformatics: a review”
IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., vol. 36, no. 5, pp. 601-615, 2006

[2]K. F. Man , K. S. Tang and S. Kwong
“Genetic algorithms: Concepts and applications”,
IEEE Trans. on Industrial Electronics, vol. 43, no. 5, pp.519 -534 1996

[3]T. Mantere and J.T. Alander
“Evolutionary Software Engineering, a Review”
Applied Soft Computing, vol. 5, no. 3, pp. 315-331, 2005

[4]Hong Guo , Asoke K. Nandi,
“Breast cancer diagnosis using genetic programming generated feature”,
Pattern Recognition, v.39 n.5, p.980-987, May, 2006

[5]H.-G. Beyer and H.-P. Schwefel.
“Evolution Strategies: A Comprehensive Introduction.”
Journal Natural Computing, 1(1):3–52, 2002.

[6]Bonabeau, E., Dorigo, M. and Theraulaz, G.
Swarm intelligence.
Oxford University Press, 1999.

[7]Kennedy, J.; Eberhart, R. (1995).
"Particle Swarm Optimization".
Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942–1948.

[8]M. Dorigo , M. Birattari and T. Stutzle
"Ant colony optimization",
IEEE Comput. Intel. Mag., vol. 1, pp.28 -39 2006
(poner tamb la del general)

[9]Karaboga, D. Basturk, B. (2007)
“A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm”
November 2007, Volume 39, Issue 3. pp 459-471 Journal of Global Optimization

[10]Mingyan Jiang, Yongming Cheng, Dongfeng Yuan.
Improved Artificial Fish Swarm Algorithm,
Proceedings of the 5th International Conference on Natural Computation (ICNC'09), 2009, 14-16 August,
Tianjin China.

[11]X. Li, Z. Shao, J. Qian, An optimizing method base on autonomous animates: fish- swarm algorithm, Systems Engineering Theory and Practice 22 (2002) 32–38.

[12]Swagatam Das, Arijit Biswas, Sambarta Dasgupta, Ajith Abraham
Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications
Foundations of Computational Intelligence Volume 3 Studies in Computational Intelligence Volume 203, 2009,
pp 23-55

[13]D. Dasgupta,
Artificial Immune Systems and Their Applications,
Springer, Berlin, 1999

[14]de Castro, L.N. and Timmis, (2002)
Artificial Immune Systems: A Novel Approach to Pattern Recognition
Artificial Neural Networks in Pattern Recognition pp. 67-84

[15]Iztok Fister, Iztok Fister, Xin-She Yang, Janez Brest
A comprehensive review of firefly algorithms
Swarm and Evolutionary Computation, Volume 13, December 2013, pp. 34–46

[16]S. He, Q. H. Wu, and J. R. Saunders,
“Group search optimizer: An optimization algorithm inspired by animal searching behavior,”
IEEE Trans. Evol. Comput., vol. 13, no. 5, pp. 973–990, Oct. 2009.

[17]HanNing Chen, Yunlong Zhu, KunYuan Hu, Tao Ku
“PS2O: A multi-swarm optimizer for discrete optimization”
Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on, pp. 587 – 592, 25-27 June 2008

[18]A.R. Mehrabian, C. Lucas
"A novel numerical optimization algorithm inspired from weed colonization",
Ecol. Inf., vol. 1, no. 4, pp.355 -366, 2006

[19]D.Simon,
“Biogeography-based optimization,”
IEEE Trans. on Evo. Com. vol.12,pp.702-713, 2008

[20]Gravel, M., Price, W.L., Gagné, C.
“Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic”
(2002) European Journal of Operational Research, 143 (1), pp. 218-229.

[21]Hani, Y., L. Amodeo, F. Yalaoui, and H. Chen.
“Ant Colony Optimization for Solving an Industrial Layout Problem.”
European Journal of Operational Research 183 (2): 633–642, 2007.

[22]Upika Premaratne , Jagath Samarabandu, and Tarlochan Sidhu,
“A New Biologically Inspired Optimization Algorithm”
,Fourth International Conference on Industrial and Information Systems, ICIIS 2009,28-31 December 2009, Sri Lanka.

[23]Shah Hosseini, H. Shahid Beheshti Univ., Tehran,
Problem solving by intelligent water drops
IEEE Congress on Evolutionary Computation, 2007. CEC 2007.

[24] Eusuff MM and K.E Lansey;
Optimization of water distribution network design using SFLA
J. Water Resour. Plann. Manage., 129(3), 210–225. (2003)