

# **Estudio y Análisis de Metaheurísticas Modernos para la Selección de Características**

**Titulación:**

Grado en Ingeniería Informática.

**Autor:**

Miguel García López

**Director:**

Daniel Molina Cabrera

---

<https://github.com/Migue8gl/>

TFG-Wrapper-Based-Metaheuristics-Feature-Selection



**UNIVERSIDAD  
DE GRANADA**

---

**ETSIIT**

Escuela Técnica Superior  
de Ingenierías Informática  
y de Telecomunicación



# Índice

## 1 Introducción

- Contexto
- Motivación
- Búsquedas Scopus
- Objetivos
- Metaheurísticas

## 2 Planificación

- Presupuesto y planificación

## 3 Revisión de la literatura

## ● Algoritmos

## 4 Diseño Experimental

- Conjuntos de datos

## 5 Resultados y Análisis

- Preguntas de Investigación
- Continuos
- Binarios
- Comparación de resultados

## 6 Conclusiones

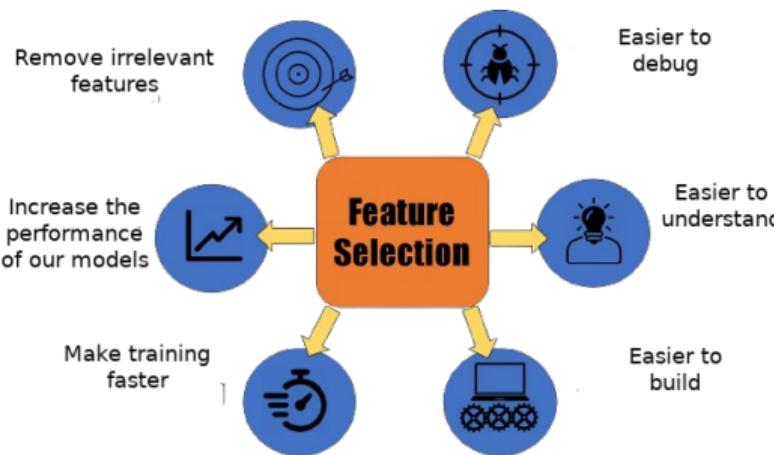
# Introducción a la Selección de Características

1 La selección de características es un proceso crucial en el aprendizaje automático

- Implica elegir un subconjunto de características relevantes.
- Es un problema **NP duro**

2 Importancia de la reducción de dimensionalidad:

- Mejora la generalización y precisión del modelo
- Reduce el ruido en los datos



Beneficios de la selección de características

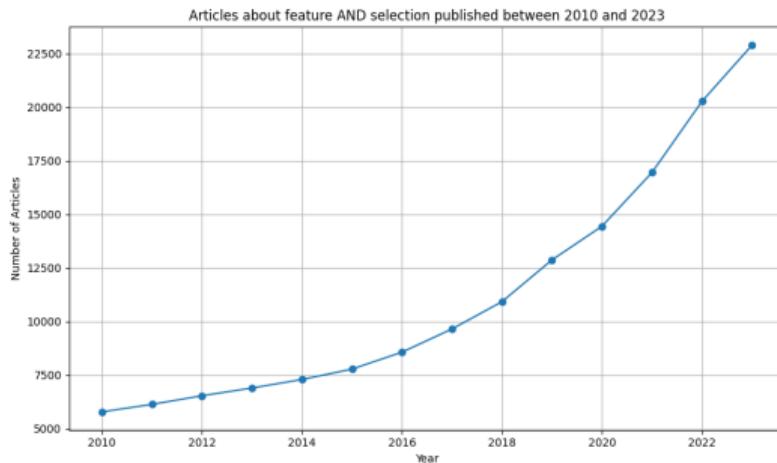
## Motivación

- 1 La selección de características es un procesamiento esencial en el aprendizaje automático
  - 2 Muchos algoritmos y métodos abordar el problema. Nos centramos en metaheurísticas
  - 3 No hay gran número de comparaciones entre versiones binarias y continuas
  - 4 Poca rigurosidad en comparaciones existentes

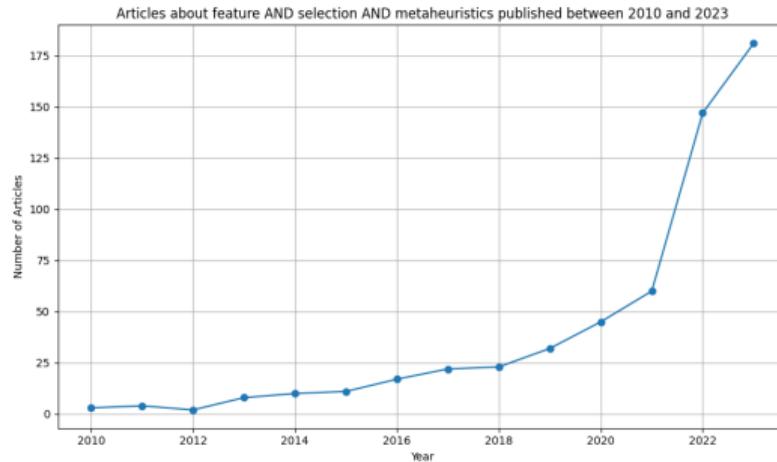
# Motivación

- ① Identificamos propuestas novedosas (problemas continuos)
- ② Implementamos versiones binarias de las anteriores
- ③ Comparación rigurosa entre algoritmos:
  - Distintas métricas
  - Test estadísticos

# Tendencia Scopus



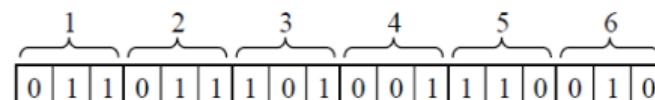
Tendencia en artículos publicados de **feature selection** en Scopus. Se incrementa exponencialmente con el tiempo.



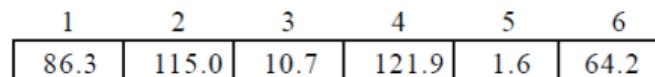
Tendencia en artículos publicados sobre **feature selection** usando **metaheurísticas** en Scopus.

## Objetivos

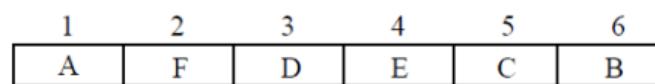
- 1 Evaluar el desempeño de las metaheurísticas
  - 2 Investigar versiones continuas y binarias
  - 3 Fortalezas y debilidades de las metaheurísticas
  - 4 Recomendaciones prácticas según el contexto
  - 5 Evaluación de resultados finales y realizar comparaciones
  - 6 Resolver una serie de preguntas de investigación. ¿Binarios vs. continuos?



### (a) Binary Encoding



### (b) Real Number Encoding



### (c) Symbol Encoding

## Codificaciones binarias y continuas<sup>1</sup>

<sup>1</sup>Shing-Tai Pan, Chih-Hung Wu, Chen-Sen Ouyang y Ying-Wei Lee. «Emotion Recognition from Speech Signals by Using Evolutionary Algorithm and Empirical Mode Decomposition». En: jul. de 2018, págs. 140-147. DOI: 10.14236/ewic/EVA2018.29

# ¿Qué son las metaheurísticas?

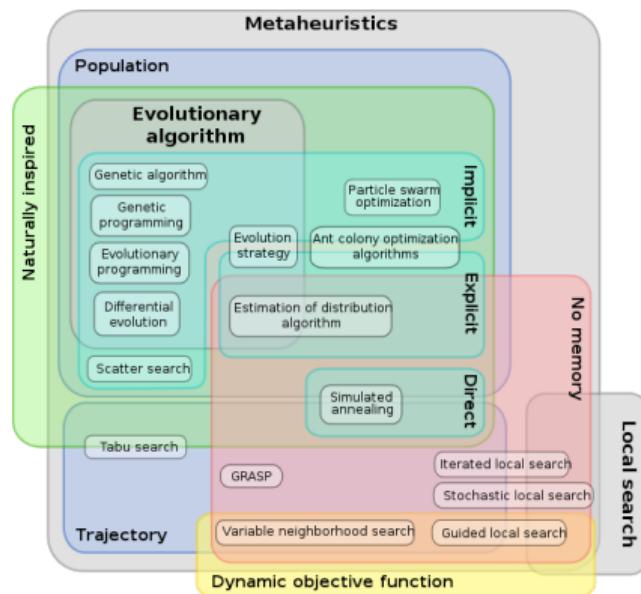
- 1 Las metaheurísticas son algoritmos de optimización
  - 2 Suelen estar bioinspiradas
  - 3 Son algoritmos generales que no dependen de un dominio en un problema específico
  - 4 Consiguen soluciones muy buenas en tiempos razonables

Algoritmo PSO en busca del **mínimo** de una función<sup>2</sup>

<sup>2</sup>Wikipedia contributors. *Particle swarm optimization* – Wikipedia, The Free Encyclopedia. [Online; accessed 26-June-2024]. 2024. URL: [https://en.wikipedia.org/wiki/Particle\\_swarm\\_optimization](https://en.wikipedia.org/wiki/Particle_swarm_optimization)



# Tipos de metaheurísticas



## Tipos de metaheurísticas

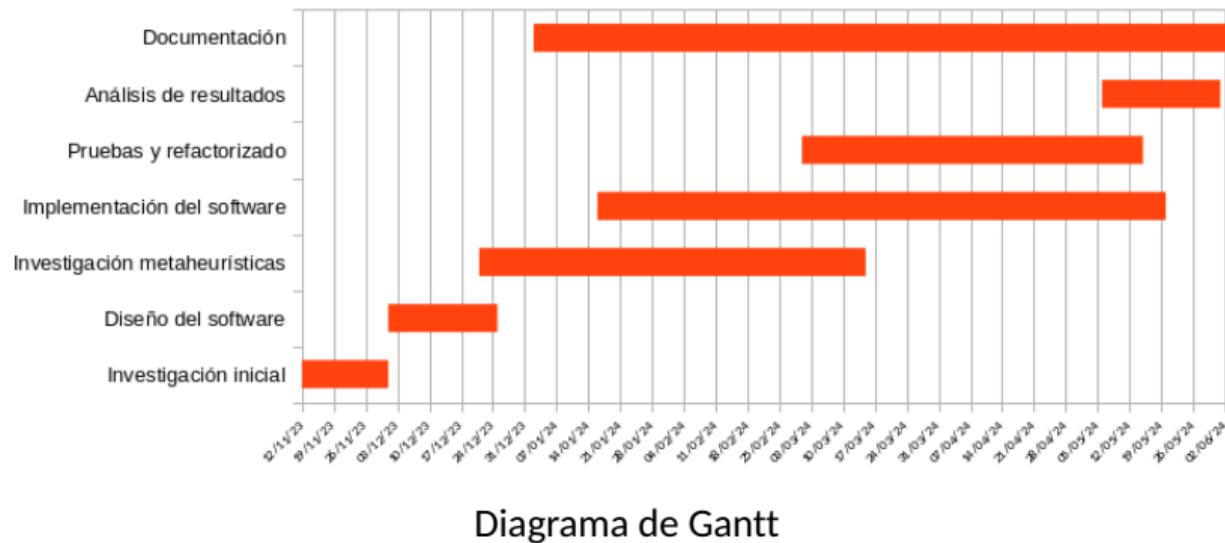
- 1 Las metaheurísticas seleccionadas en este proyecto son todas del tipo **poblacional** y evolutivas

# Tareas

Tarea	Duración Prevista (h)	Duración Final (h)
Investigación inicial	20	20
Diseño del software	10	10
Investigación metaheurísticas	60	60
Implementación del software	70	90
Pruebas y refactorizado	15	25
Ánálisis de resultados	40	20
Documentación	90	120

## Tabla de duración de cada tarea

## Diagrama de Gantt



# Presupuesto

- **Sueldo:** 25€/hora, 340 horas, total 8,500€
  - **Portátil:** HP Pavilion, 1000€, usado 8 meses, costo 133,3€
  - **Servidor:** Uso del servidor **Hércules**, costo real cero. Estimación en Google Cloud de 75,84€

# Presupuesto

Item	Costo (€)
Salario	8500
Ordenador portátil	133.3
Servidor CPU - GC Compute Engine	75.84
<b>Total</b>	<b>8709.14</b>

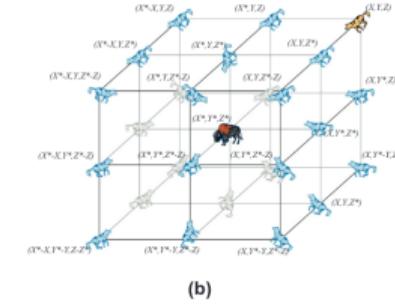
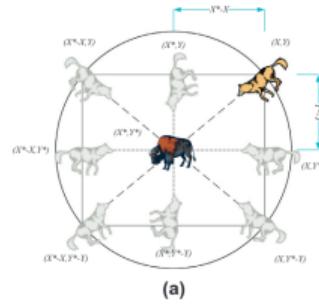
## Costo estimado del proyecto

# Algoritmos seleccionados

- 1 Se escogen para el proyecto una serie de algoritmos basándose en la investigación de aquellos más novedosos, con mejor rendimiento y más citados. Estos son los que se denominarán **modernos**
- 2 Además de los algoritmos más novedosos, se incluyen una serie de algoritmos clásicos, cuyo robustez a lo largo de los años tras multitud de aplicaciones en problemas es notable. Esta categoría, es la de los algoritmos **clásicos**

# Grey Wolf Optimizer

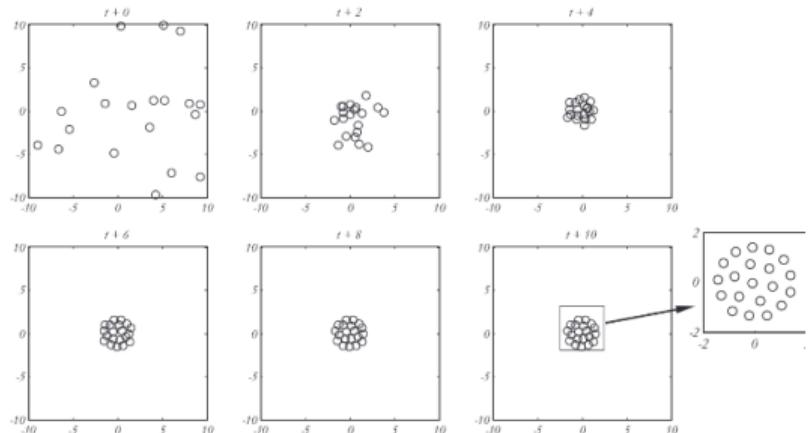
- 1 Inspirado en el comportamiento social y la técnica de caza de los lobos grises
  - 2 Modela una **jerarquía social** para guiar la búsqueda de soluciones óptimas, donde los lobos alfa, beta y delta lideran el proceso de exploración y explotación



## Caza de los lobos grises<sup>4</sup>

<sup>4</sup>S. Mirjalili, S. M. Mirjalili y A. Lewis. «Grey Wolf Optimizer». en-US. En: *Advances in Engineering Software* 69 (mar. de 2014), págs. 46-61. ISSN: 0965-9978. DOI: 10.1016/j.advengsoft.2013.12.007. URL: <https://www.sciencedirect.com/science/article/pii/S0965997813001853> (visitado 18-11-2023)

# Grasshopper Optimization Algorithm



## Convergencia de los saltamontes<sup>5</sup>

- 1 Simula el movimiento y la interacción de los saltamontes en sus distintas etapas de vida

<sup>5</sup>S. Saremi, S. Mirjalili y A. Lewis. «Grasshopper Optimisation Algorithm: Theory and application». en-US. En: *Advances in Engineering Software* 105 (mar. de 2017), págs. 30-47. ISSN: 0965-9978. DOI: 10.1016/j.advengsoft.2017.01.004. URL: <https://www.sciencedirect.com/science/article/pii/S0965997816305646> (visitado 04-11-2023)



# Firefly Algorithm

- 1 Inspirado en el comportamiento de **parpadeo y atracción** de las luciérnagas
- 2 Utiliza la intensidad de la luz como guía para la atracción entre luciérnagas, donde las luciérnagas menos brillantes se mueven hacia las más brillantes



Imagen de una libélula con su característico brillo

# Cuckoo Search



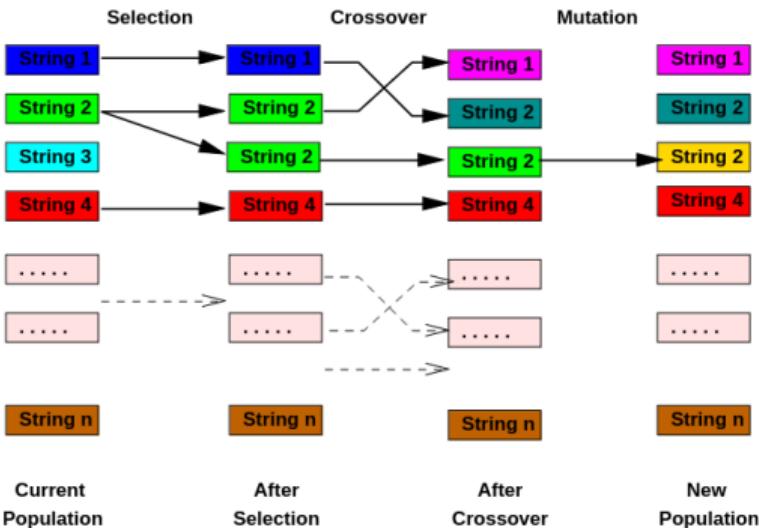
Cuatro nidos de huevos de pájaro. En cada uno de ellos un huevo visiblemente más grande del pájaro Cuco<sup>6</sup>

- ➊ Inspirado en el comportamiento de anidación de los cucos y el **parasitismo** de puesta
- ➋ Caracterizado por usar métodos de búsqueda aleatoria y el método **Levy flight** para la exploración del espacio de soluciones

<sup>6</sup>Chiswick Chap. Cuckoo Eggs Mimicking Reed Warbler Eggs. CC BY-SA 3.0. 2024. URL: <https://commons.wikimedia.org/w/index.php?curid=18861052>

# Genetic Algorithm

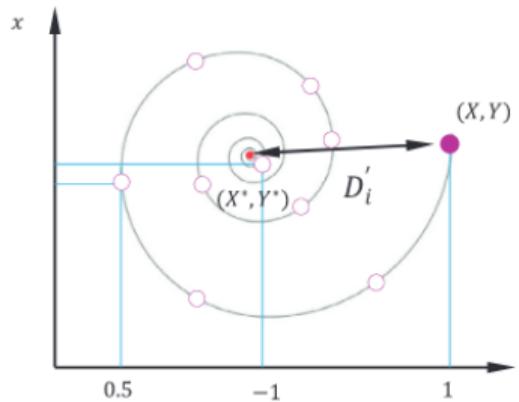
- 1 Algoritmo basado en la recombinación de cromosomas, que toma inspiración de la **evolución** biológica y genética
- 2 Hace uso de operadores tales como la **mutación** o el **cruce**



Principio básico del algoritmo GA <sup>7</sup>

<sup>7</sup>T. V. Mathew. «Genetic algorithm». En: *Report submitted at IIT Bombay 53* (2012)

# Whale Optimization Algorithm



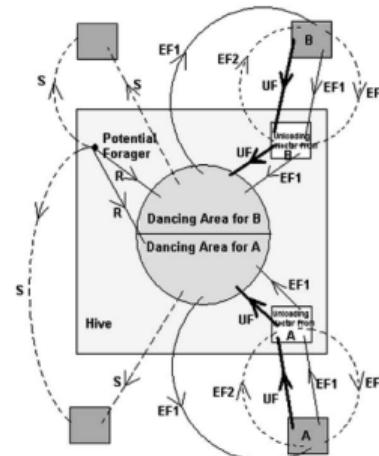
- 1 Inspirado en el comportamiento de las ballenas jorobadas
  - 2 Usa principalmente dos variantes de operadores de caza:
    - Espiral de búsqueda
    - Técnica de burbujeo de red

Espiral para simular el mecanismo de ataque de la red de burbujas de las ballenas jorobadas<sup>8</sup>

<sup>8</sup>S. Mirjalili y A. Lewis. «The Whale Optimization Algorithm». en-US. En: *Advances in Engineering Software* 95 (mayo de 2016), págs. 51-67. ISSN: 0965-9978. DOI: 10.1016/j.advengsoft.2016.01.008. URL: <https://www.sciencedirect.com/science/article/pii/S0965997816300163> (visitado 14-10-2023)

# Artificial Bee Colony Optimization

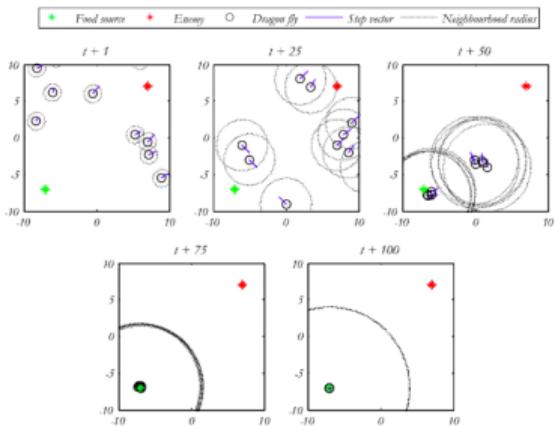
- 1 Simula la **búsqueda de alimentos** de las abejas empleadas, las abejas observadoras y las abejas exploradoras para encontrar soluciones óptimas



## Diagrama de funcionamiento del ABCO<sup>9</sup>

<sup>9</sup>D. Karaboga y B. Akay. «A comparative study of Artificial Bee Colony algorithm». En: *Applied Mathematics and Computation* 214.1 (2009). Cited by: 2940, págs. 108-132. DOI: 10.1016/j.amc.2009.03.090. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-67349273050&doi=10.1016%2fj.amc.2009.03.090&partnerID=40&md5=505464030a4a96a1998b20803cd113ce>

# Dragonfly Algorithm



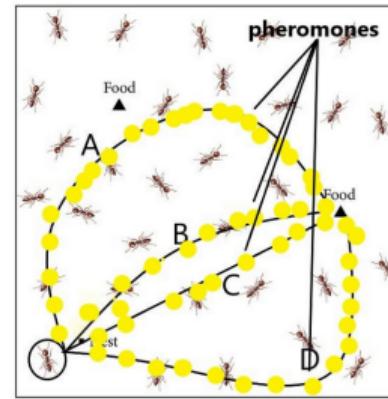
Operadores del algoritmo DA<sup>10</sup>

- 1 Basado en el comportamiento de enjambre y formación de las libélulas, usando operadores que controlan características como la **cohesión** de grupo o **distanciamiento** del enemigo, entre otros
- 2 Simula las interacciones sociales y el movimiento de las libélulas para equilibrar la exploración y explotación del espacio de soluciones

<sup>10</sup>Yassine Meraihi, Amar Ramdane-Cherif, Dalila Acheli y Mohammed Mahseur. «Dragonfly algorithm: a comprehensive review and applications». En: *Neural Computing and Applications* 32.21 (nov. de 2020), págs. 16625-16646. ISSN: 1433-3058. DOI: 10.1007/s00521-020-04866-y. URL: <https://doi.org/10.1007/s00521-020-04866-y>

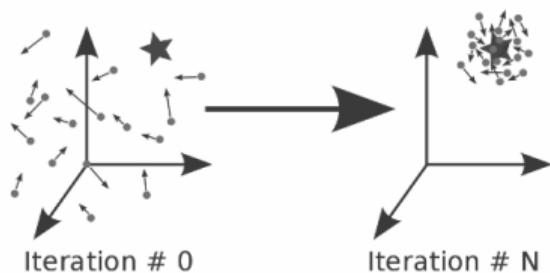
# Ant Colony Optimization

- 1 Simula las colonias de hormigas. Para ello usa el rastro de **feromonas** para guiar la búsqueda de soluciones óptimas, donde las hormigas depositan y siguen feromonas en los caminos más prometedores



Caminos de un grafo marcados por la feromona,  
operador esencial de ACO

# Particle Swarm Optimization

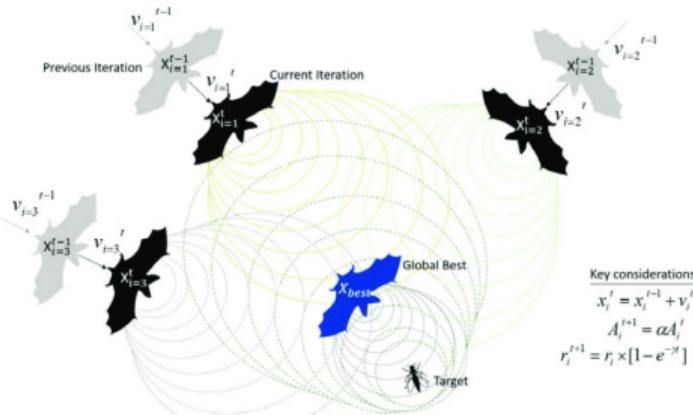


Partículas en el espacio (con una velocidad y dirección) convergiendo en la iteración  $N$

- ① Inspirado en el comportamiento social de los enjambres de aves y peces
- ② Simula la búsqueda colectiva de soluciones, donde cada partícula ajusta su posición basada en su **experiencia** personal y la de sus **vecinos**

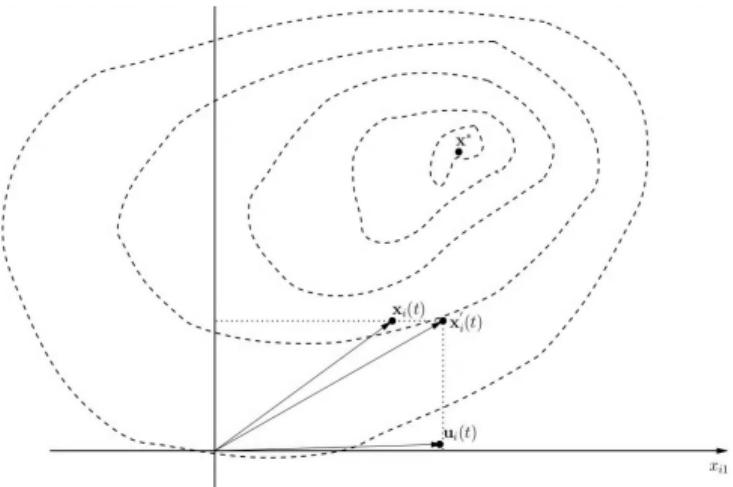
## Bat Algorithm

- ① Basado en el comportamiento de **ecolocalización** de los murciélagos
  - ② Utiliza la técnica de emisión de pulsos y el ajuste de frecuencia para explorar y explotar el espacio de soluciones



## Funcionamiento del algoritmo BA

# Differential Evolution



- 1 Utiliza la combinación y mutación de vectores solución para buscar la mejor solución, enfocándose en la **diferencia** entre las soluciones actuales para generar nuevas

Operador de crossover o cruce de DE <sup>11</sup>

<sup>11</sup>A. P. Engelbrecht. *Computational Intelligence: An Introduction*. 2nd. Wiley Publishing, 2007. ISBN: 0470035617

## Conjuntos de datos

- 1 Se escogen conjuntos de datos por:
    - Variedad de áreas
    - Diversidad de problemas
    - Número de características
    - Relevancia práctica

Dataset	Inst.	Car.	Cls.	Área
sonar	207	60	2	Biología
spambase-460	459	54	2	Informática
spectf-heart	348	44	2	Medicina
waveform5000	5000	40	3	Física
ionosphere	350	34	2	Meteorología
dermatology	366	34	6	Medicina
wdbc	568	29	2	Medicina
parkinsons	200	22	2	Medicina
zoo	101	18	7	Biología
wine	182	13	3	Alimentación
breast-cancer	286	9	2	Medicina
diabetes	768	8	2	Medicina
yeast	1483	8	10	Biología
ecoli	336	7	8	Biología
iris	149	4	3	Biología

## Información de conjuntos de datos por número de características

## Función fitness

- ① La función *fitness* se construye con:

- Accuracy al 90 %
  - Reducción al 10 %

- 2 Se define como:

$$\text{fitness} = \text{acc} \cdot \alpha + \text{red} \cdot (1 - \alpha) \quad (1)$$

Donde  $\alpha$  es la ponderación dada a la precisión o accuracy

- ③ Para el cálculo de precisión se usan los clasificadores kNN y SVC.

# Parámetros de los algoritmos

Algoritmo	Parámetros
GOA	$c_{min}$ : 0.00001 $c_{max}$ : 1 F: 0.5 L: 1.5
WOA	Parámetro espiral: 1
ABCO	Abeja empleada: 3 Abeja vigilante: 3 Límite: 3
BA	$\alpha$ : 0.9 $\gamma$ : 0.9 $f_{min}$ : 0 $f_{max}$ : 2
PSO	w: 0.9 $c_1$ : 2 $c_2$ : 2
FA	$\alpha_0$ : 0.5 $\beta_0$ : 0.2 $\gamma_0$ : 1
GA	Ratio de cruce: 1 Ratio de mutación: 0.05 Elite: 2 $\eta$ : 1 $\alpha$ : $\sqrt{0,3}$
ACO	$\alpha$ : 1 Q: 1 Feromona inicial: 0.1 Ratio de evaporación: 0.049
CS	Ratio de descubrimiento: 0.25 $\alpha$ : 1 $\lambda$ : 1.5
DE	F: 0.5 Cr: 0.1

Parámetros de diferentes algoritmos de optimización

# Comparativas

- ① Se muestran resultados de los experimentos
- ② Hay muchas más comparativas en la sección del apéndice en la memoria principal:
  - Figuras de convergencia
  - Figuras de boxplots
  - Tablas comparativas en todas las métricas (*fitness*, *accuracy*, reducción, tiempo de ejecución, etc.)
  - Tablas con resultados de los test estadísticos

# Preguntas de Investigación

- ¿Merece la pena el uso de algoritmos específicos para la selección de características?
- ¿Cómo se comparan los algoritmos recientes con los clásicos?
- ¿Cuáles de los algoritmos recientes parecen más prometedores?
- ¿Son los algoritmos igualmente eficaces en su versión binaria y continua?
- ¿Cuáles son las opciones más interesantes en ciertos contextos?

# Preguntas de Investigación

- ¿Merece la pena el uso de algoritmos específicos para la selección de características?
- ¿Cómo se comparan los algoritmos recientes con los clásicos?
- ¿Cuáles de los algoritmos recientes parecen más prometedores?
- ¿Son los algoritmos igualmente eficaces en su versión binaria y continua?
- ¿Cuáles son las opciones más interesantes en ciertos contextos?

# Preguntas de Investigación

- ¿Merece la pena el uso de algoritmos específicos para la selección de características?
- ¿Cómo se comparan los algoritmos recientes con los clásicos?
- ¿Cuáles de los algoritmos recientes parecen más prometedores?
- ¿Son los algoritmos igualmente eficaces en su versión binaria y continua?
- ¿Cuáles son las opciones más interesantes en ciertos contextos?

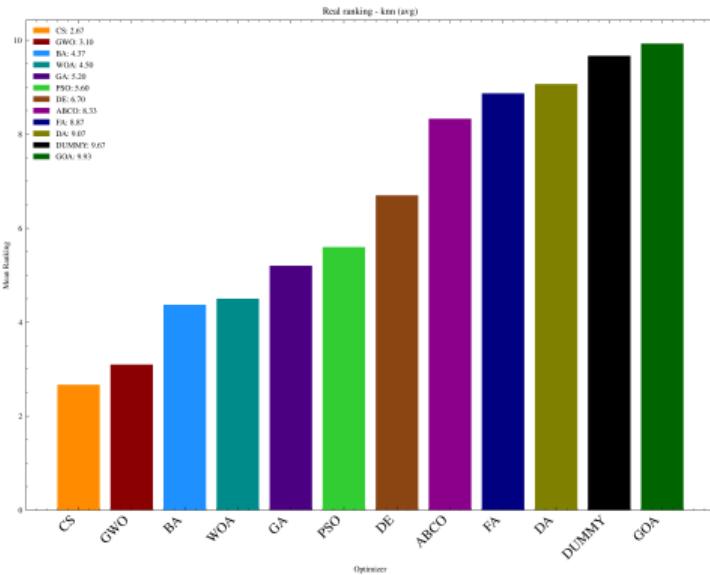
# Preguntas de Investigación

- ¿Merece la pena el uso de algoritmos específicos para la selección de características?
- ¿Cómo se comparan los algoritmos recientes con los clásicos?
- ¿Cuáles de los algoritmos recientes parecen más prometedores?
- ¿Son los algoritmos igualmente eficaces en su versión binaria y continua?
- ¿Cuáles son las opciones más interesantes en ciertos contextos?

# Preguntas de Investigación

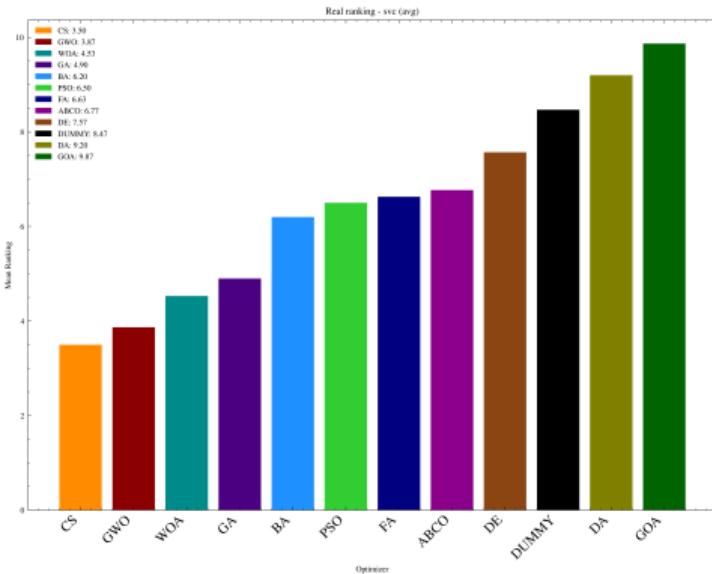
- ¿Merece la pena el uso de algoritmos específicos para la selección de características?
- ¿Cómo se comparan los algoritmos recientes con los clásicos?
- ¿Cuáles de los algoritmos recientes parecen más prometedores?
- ¿Son los algoritmos igualmente eficaces en su versión binaria y continua?
- ¿Cuáles son las opciones más interesantes en ciertos contextos?

# Ranking en continuos para fitness: kNN



Ranking de los algoritmos en versión continua para kNN

# Ranking en continuos para fitness: SVC



Ranking de los algoritmos en versión continua para SVC

# Resultados en continuos

- Los mejores algoritmos en *fitness* son:
  - CS (Cuckoo Search)
  - GWO (Grey Wolf Optimizer)
- Los peores algoritmos son:
  - GOA (Grasshopper Optimization Algorithm)
  - DA (Dragonfly Algorithm)
- Mejor rendimiento en reducción de características:
  - CS y GWO (con mucha diferencia)
- Eficiencia temporal:
  - Más rápido: FA (Firefly Algorithm)
  - Más lento: ABCO (Artificial Bee Colony Optimization)

# Resultados en continuos

- Los mejores algoritmos en *fitness* son:
  - CS (Cuckoo Search)
  - GWO (Grey Wolf Optimizer)
- Los peores algoritmos son:
  - GOA (Grasshopper Optimization Algorithm)
  - DA (Dragonfly Algorithm)
- Mejor rendimiento en reducción de características:
  - CS y GWO (con mucha diferencia)
- Eficiencia temporal:
  - Más rápido: FA (Firefly Algorithm)
  - Más lento: ABCO (Artificial Bee Colony Optimization)



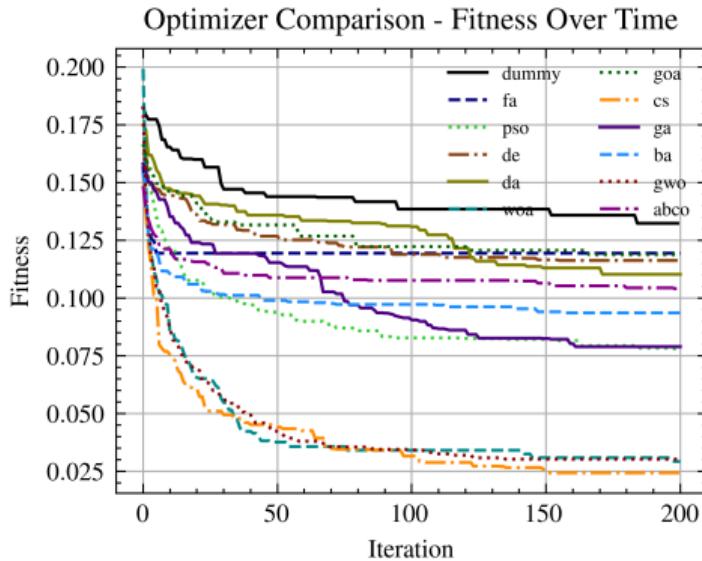
# Resultados en continuos

- Los mejores algoritmos en *fitness* son:
  - CS (Cuckoo Search)
  - GWO (Grey Wolf Optimizer)
- Los peores algoritmos son:
  - GOA (Grasshopper Optimization Algorithm)
  - DA (Dragonfly Algorithm)
- Mejor rendimiento en reducción de características:
  - CS y GWO (con mucha diferencia)
- Eficiencia temporal:
  - Más rápido: FA (Firefly Algorithm)
  - Más lento: ABCO (Artificial Bee Colony Optimization)

# Resultados en continuos

- Los mejores algoritmos en *fitness* son:
  - CS (Cuckoo Search)
  - GWO (Grey Wolf Optimizer)
- Los peores algoritmos son:
  - GOA (Grasshopper Optimization Algorithm)
  - DA (Dragonfly Algorithm)
- Mejor rendimiento en reducción de características:
  - CS y GWO (con mucha diferencia)
- Eficiencia temporal:
  - Más rápido: FA (Firefly Algorithm)
  - Más lento: ABCO (Artificial Bee Colony Optimization)

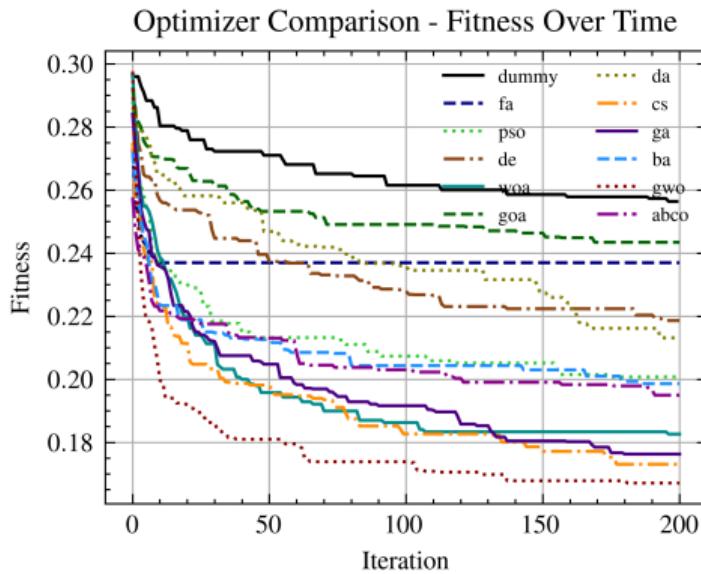
## Convergencia en continuo: Ionosphere - kNN



Convergencia de todas las metaheurísticas en ionosphere - knn - real

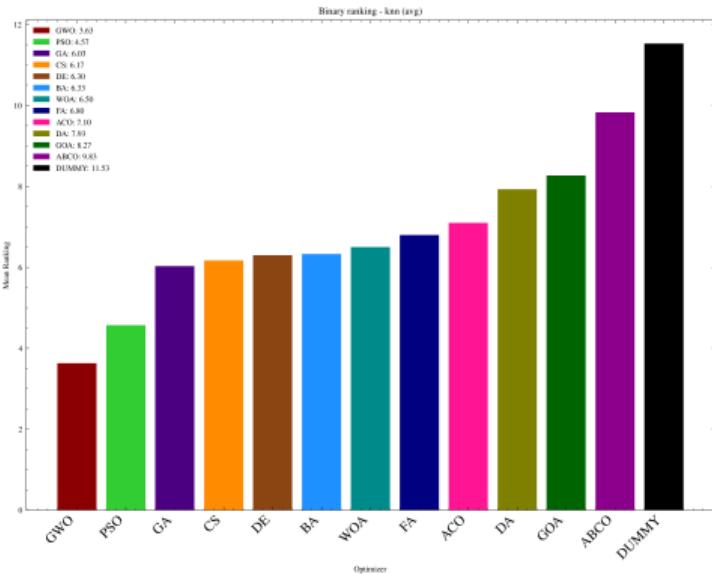


# Convergencia en continuo: Diabetes - kNN



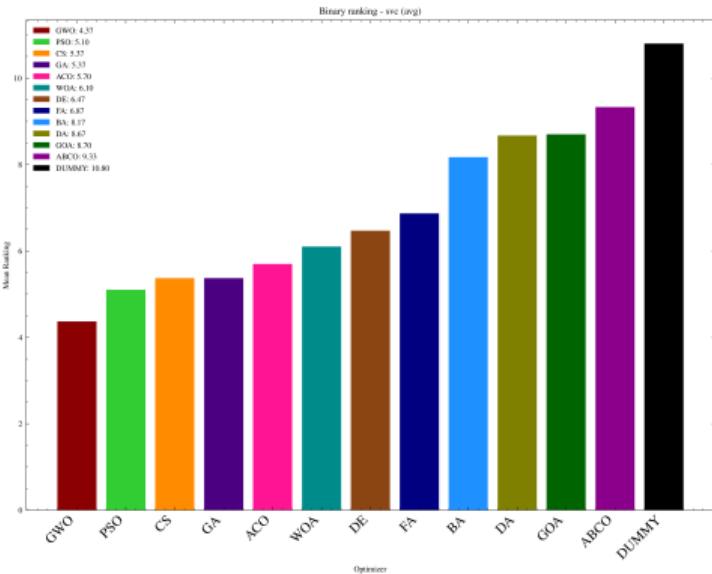
Convergencia de todas las metaheurísticas en diabetes - knn - real

# Ranking en binario para fitness: kNN



Ranking de los algoritmos en versión binaria para kNN

# Ranking en binario para fitness: SVC



Ranking de los algoritmos en versión binaria para SVC

# Resultados en binarios

- Los mejores algoritmos en *fitness* son:
  - **bGWO (Binary Grey Wolf Optimizer)**
  - **bPSO (Binary Particle Swarm Optimization)**
- Los peores algoritmos son:
  - **bGOA (Binary Grasshopper Optimization Algorithm)**
  - **bABCO (Binary Artificial Bee Colony Optimization)**
- Mejor rendimiento en reducción de características:
  - **ACO (Ant Colony Optimization)**
- Eficiencia temporal:
  - Más rápido: **bFA (Binary Firefly Algorithm)**
  - Más lento: **bABCO (Binary Artificial Bee Colony Optimization)**

# Resultados en binarios

- Los mejores algoritmos en fitness son:
  - **bGWO (Binary Grey Wolf Optimizer)**
  - **bPSO (Binary Particle Swarm Optimization)**
- Los peores algoritmos son:
  - **bGOA (Binary Grasshopper Optimization Algorithm)**
  - **bABCO (Binary Artificial Bee Colony Optimization)**
- Mejor rendimiento en reducción de características:
  - **ACO (Ant Colony Optimization)**
- Eficiencia temporal:
  - Más rápido: **bFA (Binary Firefly Algorithm)**
  - Más lento: **bABCO (Binary Artificial Bee Colony Optimization)**

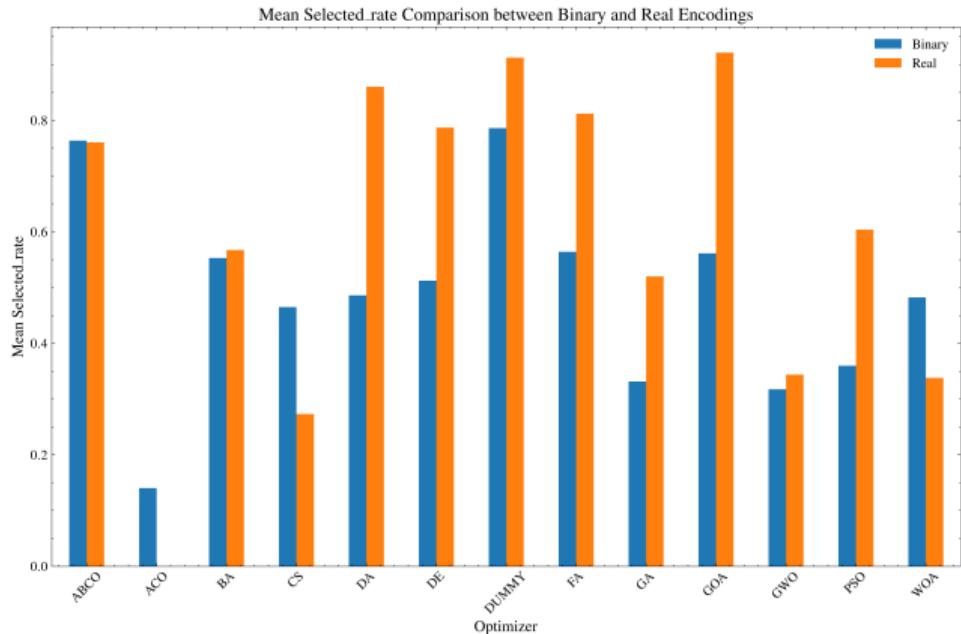
# Resultados en binarios

- Los mejores algoritmos en fitness son:
  - **bGWO (Binary Grey Wolf Optimizer)**
  - **bPSO (Binary Particle Swarm Optimization)**
- Los peores algoritmos son:
  - **bGOA (Binary Grasshopper Optimization Algorithm)**
  - **bABCO (Binary Artificial Bee Colony Optimization)**
- Mejor rendimiento en reducción de características:
  - **ACO (Ant Colony Optimization)**
- Eficiencia temporal:
  - Más rápido: **bFA (Binary Firefly Algorithm)**
  - Más lento: **bABCO (Binary Artificial Bee Colony Optimization)**

# Resultados en binarios

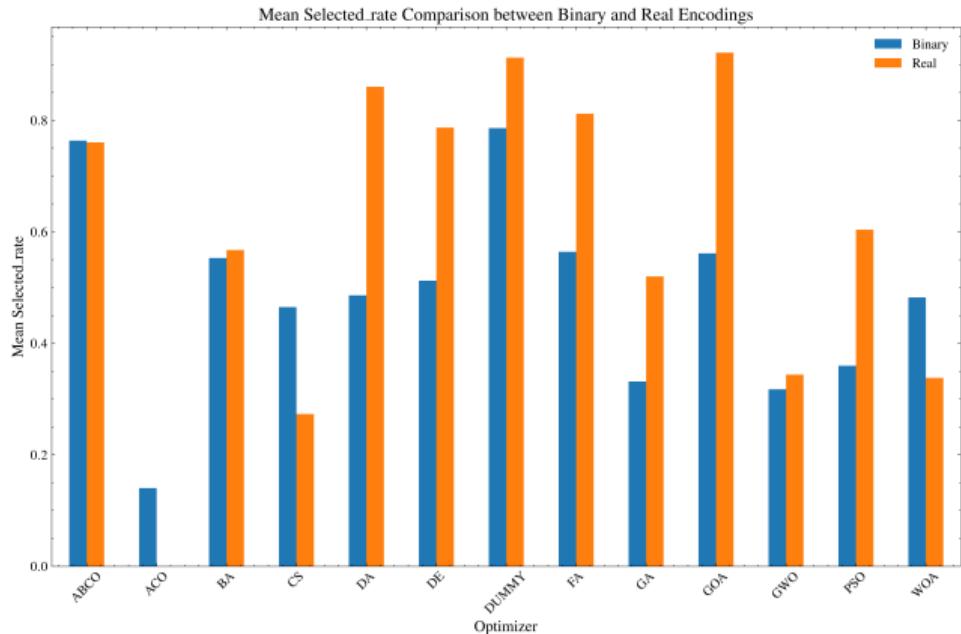
- Los mejores algoritmos en fitness son:
  - **bGWO (Binary Grey Wolf Optimizer)**
  - **bPSO (Binary Particle Swarm Optimization)**
- Los peores algoritmos son:
  - **bGOA (Binary Grasshopper Optimization Algorithm)**
  - **bABCO (Binary Artificial Bee Colony Optimization)**
- Mejor rendimiento en reducción de características:
  - **ACO (Ant Colony Optimization)**
- Eficiencia temporal:
  - Más rápido: **bFA (Binary Firefly Algorithm)**
  - Más lento: **bABCO (Binary Artificial Bee Colony Optimization)**

# Comparación de algoritmos específicos vs. originales



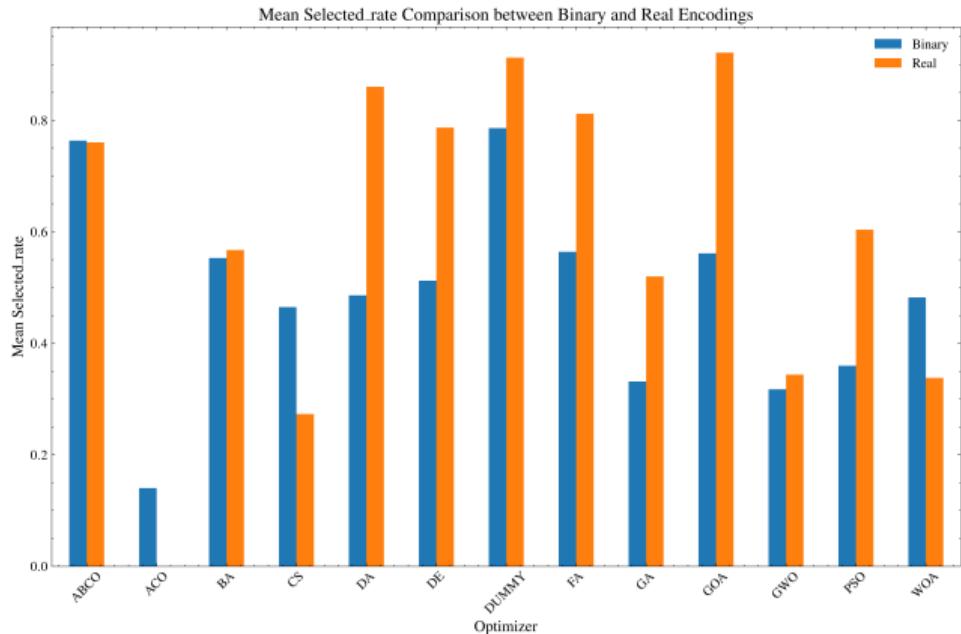
- Los algoritmos binarios tienden a seleccionar menos características
- Algoritmos continuos también capaces de reducir características
- **Conclusión:** Ambos enfoques son válidos para la selección de características, pero los binarios son más eficaces

# Comparación de algoritmos específicos vs. originales



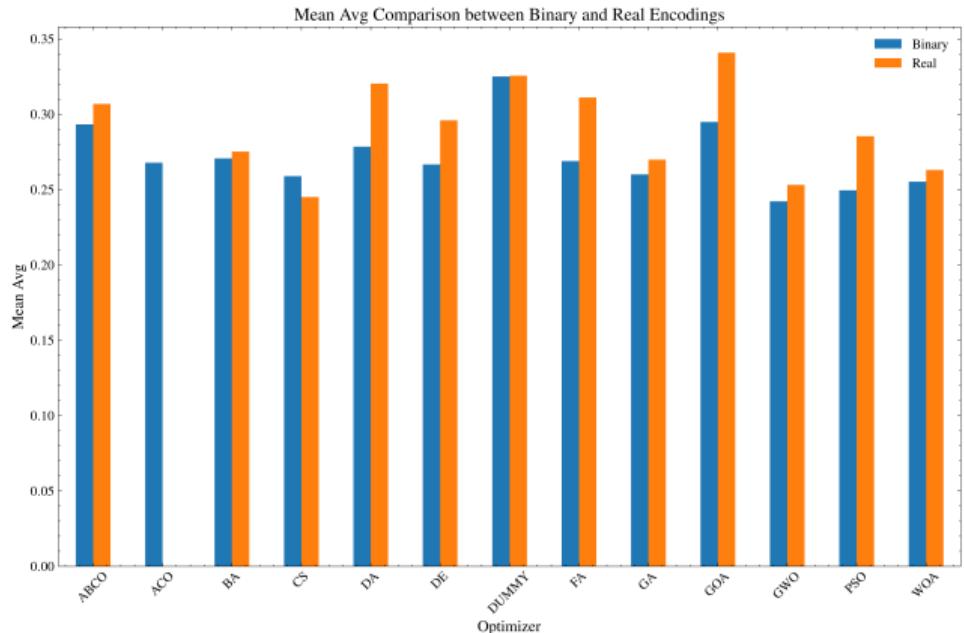
- Los algoritmos binarios tienden a seleccionar menos características
- Algoritmos continuos también capaces de reducir características
- **Conclusión:** Ambos enfoques son válidos para la selección de características, pero los binarios son más eficaces

# Comparación de algoritmos específicos vs. originales



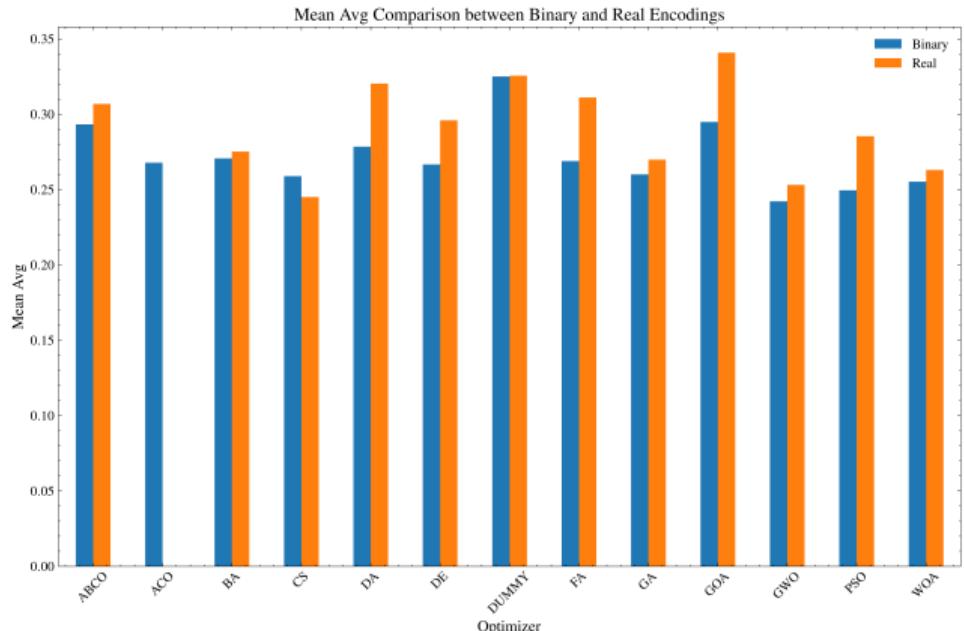
- Los algoritmos binarios tienden a seleccionar menos características
- Algoritmos continuos también capaces de reducir características
- **Conclusión:** Ambos enfoques son válidos para la selección de características, pero los binarios son más eficaces

# Comparación de rendimiento: Recientes vs. Clásicos



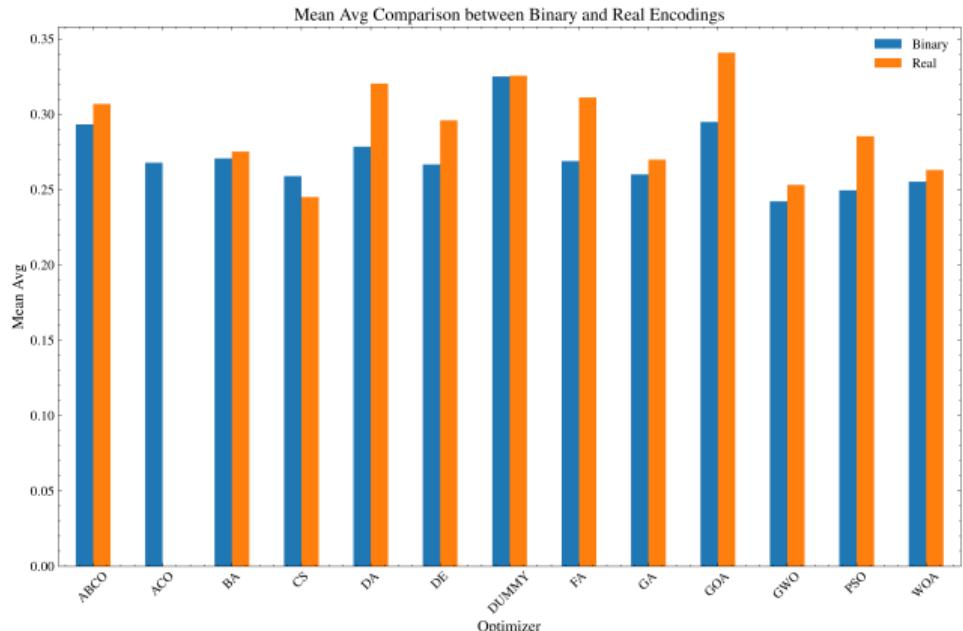
- Algoritmos recientes (GWO, CS) muestran excelente rendimiento
- Algunos clásicos (PSO, DE, GA) siguen siendo competitivos
- **Conclusión:** Algunos recientes ofrecen mejoras, pero los clásicos siguen siendo muy robustos

# Comparación de rendimiento: Recientes vs. Clásicos



- Algoritmos recientes (GWO, CS) muestran excelente rendimiento
- Algunos clásicos (PSO, DE, GA) siguen siendo competitivos
- *Conclusión:* Algunos recientes ofrecen mejoras, pero los clásicos siguen siendo muy robustos

# Comparación de rendimiento: Recientes vs. Clásicos



- Algoritmos recientes (GWO, CS) muestran excelente rendimiento
- Algunos clásicos (PSO, DE, GA) siguen siendo competitivos
- **Conclusión:** Algunos recientes ofrecen mejoras, pero los clásicos siguen siendo muy robustos

# Algoritmos recientes prometedores

- En versiones continuas:
  - GWO (Grey Wolf Optimizer)
  - CS (Cuckoo Search)
- En versiones binarias:
  - bGWO (Binary Grey Wolf Optimizer)
  - bPSO (Binary Particle Swarm Optimization)
- **Conclusión:** GWO destaca en ambas versiones, mostrando gran adaptabilidad y rendimiento

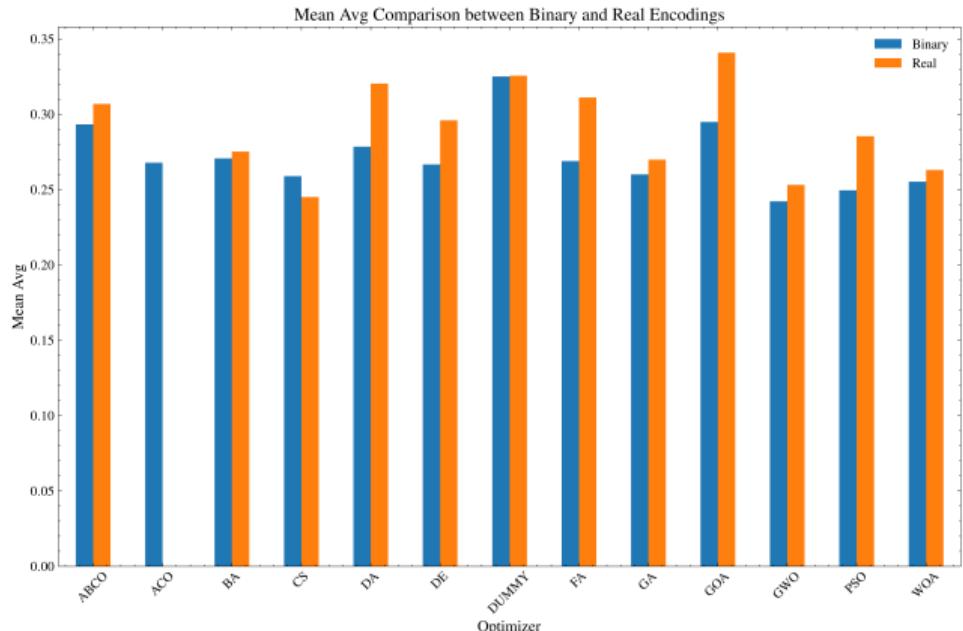
# Algoritmos recientes prometedores

- En versiones continuas:
  - **GWO (Grey Wolf Optimizer)**
  - **CS (Cuckoo Search)**
- En versiones binarias:
  - **bGWO (Binary Grey Wolf Optimizer)**
  - **bPSO (Binary Particle Swarm Optimization)**
- **Conclusión:** GWO destaca en ambas versiones, mostrando gran adaptabilidad y rendimiento

# Algoritmos recientes prometedores

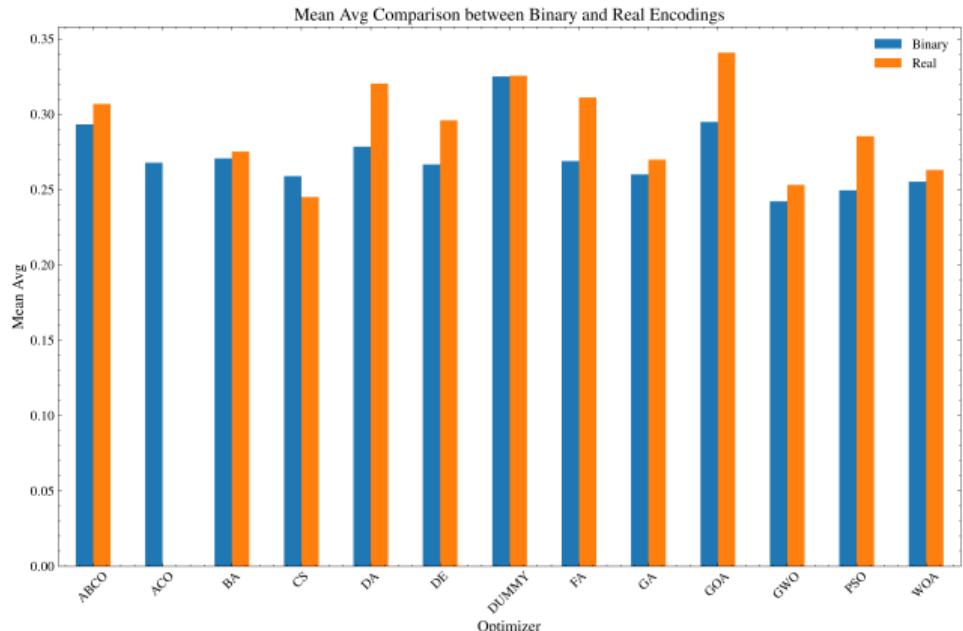
- En versiones continuas:
  - GWO (Grey Wolf Optimizer)
  - CS (Cuckoo Search)
- En versiones binarias:
  - bGWO (Binary Grey Wolf Optimizer)
  - bPSO (Binary Particle Swarm Optimization)
- **Conclusión:** GWO destaca en ambas versiones, mostrando gran adaptabilidad y rendimiento

# Eficacia en versiones originales vs. binarias



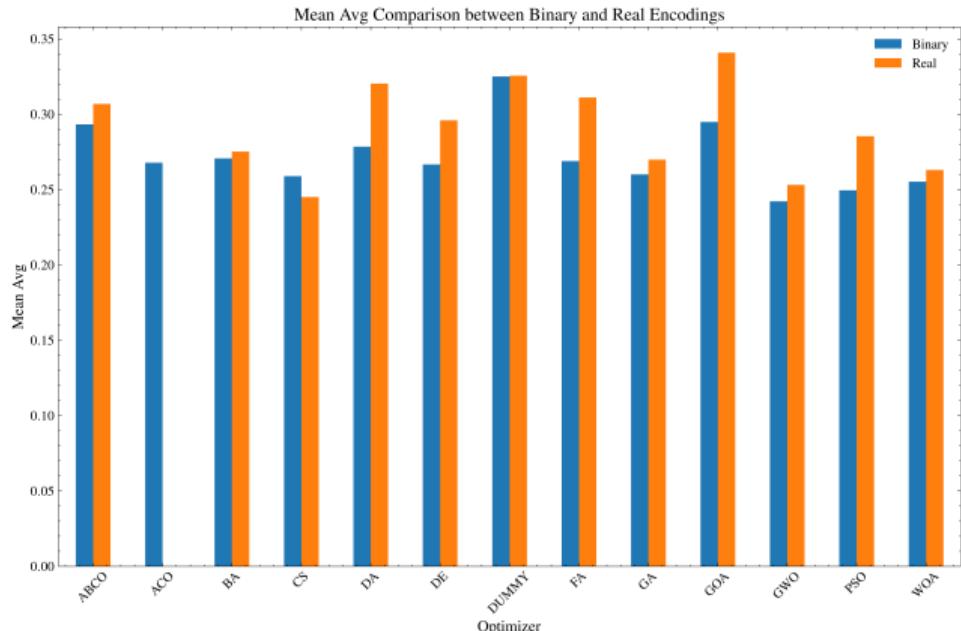
- Rendimiento generalmente similar
- Ligeras ventajas para versiones continuas en algunos casos
- **Conclusión:** La mayoría de los algoritmos mantienen su eficacia al ser adaptados a versiones binarias

# Eficacia en versiones originales vs. binarias



- Rendimiento generalmente similar
- Ligeras ventajas para versiones continuas en algunos casos
- **Conclusión:** La mayoría de los algoritmos mantienen su eficacia al ser adaptados a versiones binarias

# Eficacia en versiones originales vs. binarias



- Rendimiento generalmente similar
- Ligeras ventajas para versiones continuas en algunos casos
- **Conclusión:** La mayoría de los algoritmos mantienen su eficacia al ser adaptados a versiones binarias

# Opciones más interesantes por contexto

- Mejor fitness general:
  - Continuo: **CS, GWO**
  - Binario: **bGWO, bPSO**
- Mejor reducción de características:
  - Continuo: **CS, GWO** (con mucha diferencia)
  - Binario: **ACO**
- Mayor eficiencia temporal:
  - Continuo: **FA (Firefly Algorithm)**
  - Binario: **bFA (Binary Firefly Algorithm)**
- **Conclusión:** La elección del algoritmo depende de las prioridades específicas del problema (fitness, reducción de características, eficiencia)

# Opciones más interesantes por contexto

- Mejor fitness general:
  - Continuo: **CS, GWO**
  - Binario: **bGWO, bPSO**
- Mejor reducción de características:
  - Continuo: **CS, GWO** (con mucha diferencia)
  - Binario: **ACO**
- Mayor eficiencia temporal:
  - Continuo: **FA (Firefly Algorithm)**
  - Binario: **bFA (Binary Firefly Algorithm)**
- **Conclusión:** La elección del algoritmo depende de las prioridades específicas del problema (fitness, reducción de características, eficiencia)

# Opciones más interesantes por contexto

- Mejor fitness general:
  - Continuo: **CS, GWO**
  - Binario: **bGWO, bPSO**
- Mejor reducción de características:
  - Continuo: **CS, GWO** (con mucha diferencia)
  - Binario: **ACO**
- Mayor eficiencia temporal:
  - Continuo: **FA (Firefly Algorithm)**
  - Binario: **bFA (Binary Firefly Algorithm)**
- **Conclusión:** La elección del algoritmo depende de las prioridades específicas del problema (fitness, reducción de características, eficiencia)

# Opciones más interesantes por contexto

- Mejor fitness general:
  - Continuo: **CS, GWO**
  - Binario: **bGWO, bPSO**
- Mejor reducción de características:
  - Continuo: **CS, GWO** (con mucha diferencia)
  - Binario: **ACO**
- Mayor eficiencia temporal:
  - Continuo: **FA (Firefly Algorithm)**
  - Binario: **bFA (Binary Firefly Algorithm)**
- **Conclusión:** La elección del algoritmo depende de las prioridades específicas del problema (fitness, reducción de características, eficiencia)

# ¿Algoritmos específicos vs. originales?

- Ambos enfoques son capaces de reducir características
  - Los algoritmos binarios son mucho más eficaces

# ¿Algoritmos específicos vs. originales?

- Ambos enfoques son capaces de reducir características
  - Los algoritmos binarios son mucho más eficaces

# ¿Algoritmos recientes vs. clásicos?

- Algunos recientes (GWO, CS) muestran excelente rendimiento
- Algoritmos clásicos (PSO, GA) siguen siendo competitivos



## ¿Algoritmos recientes vs. clásicos?

- Algunos recientes (GWO, CS) muestran excelente rendimiento
  - Algoritmos clásicos (PSO, GA) siguen siendo competitivos

## ¿Cuáles son los algoritmos más prometedores?

- GWO y CS destacan en versiones continuas y binarias

## ¿Eficacia binaria vs. continua?

- Generalmente similar
  - Ligeras ventajas para versiones continuas en algunos casos y viceversa

# ¿Eficacia binaria vs. continua?

- Generalmente similar
- Ligeras ventajas para versiones continuas en algunos casos y viceversa

# ¿Qué opciones elegir según el contexto?

- Mejor fitness general:
  - Continuo: CS, GWO
  - Binario: bGWO, bPSO
- Mejor reducción de características:
  - Continuo: CS, GWO
  - Binario: ACO
- Mayor eficiencia temporal:
  - FA (tanto en versión continua como binaria)

# ¿Qué opciones elegir según el contexto?

- Mejor fitness general:
  - Continuo: CS, GWO
  - Binario: bGWO, bPSO
- Mejor reducción de características:
  - Continuo: CS, GWO
  - Binario: ACO
- Mayor eficiencia temporal:
  - FA (tanto en versión continua como binaria)

# ¿Qué opciones elegir según el contexto?

- Mejor fitness general:
  - Continuo: CS, GWO
  - Binario: bGWO, bPSO
- Mejor reducción de características:
  - Continuo: CS, GWO
  - Binario: ACO
- Mayor eficiencia temporal:
  - FA (tanto en versión continua como binaria)

## Agradecimientos

Gracias por su atención.

¿Dudas, preguntas o comentarios?

