University of Granada

Department of Computer Science
and Artificial Intelligence

PhD Program in Information and
Communication Technologies

PhD THESIS DISSERTATION

Metaheuristics for the Design of Deep Learning Models

PhD CANDIDATE
Javier Poyatos Amador

PhD ADVISORS
Francisco Herrera Triguero & Daniel Molina Cabrera

Granada, April 2024

Editor: Universidad de Granada. Tesis Doctorales
Autor: Javier Poyatos Amador
ISBN: 978-84-1195-387-0
URI: https://hdl.handle.net/10481/93085

El doctorando / *The PhD candidate* **Javier Poyatos Amador** y los directores / *and the advisors* **Francisco Herrera Triguero & Daniel Molina Cabrera**

Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y, hasta donde nuestro conocimiento alcanza, en la realización del trabajo se han respetado los derechos de otros autores a ser citados cuando se han utilizado sus resultados o publicaciones.

Guarantee, by signing this doctoral thesis, that the work has been done by the doctoral candidate under the direction of the thesis supervisors and, as far as our knowledge reaches, the rights of the authors to be cited (when their results or publications have been used) have been respected in the performance of this work.

Granada, April 2024.

The PhD candidate:

Sgd.: Javier Poyatos Amador

The PhD advisor: The PhD advisor:

Sgd.: Francisco Herrera Triguero Sgd.: Daniel Molina Cabrera

Funding

This doctoral thesis has been funded by the predoctoral scholarship granted by the University of Granada.

«*Mathematics is a more powerful instrument of knowledge than any other that has been bequeathed to us by human agency.*».

- René Descartes.

«*Do or Do Not, there is no try.*».

- Master Yoda.
The Empire Strikes Back, Star Wars.

Agradecimientos

Hace ya cuatro años que me embarqué en el reto de realizar una tesis doctoral. Estos años se pueden resumir como una montaña rusa de emociones positivas y negativas donde las ejecuciones exitosas y fallidas marcaban el paso de los días. Llegaron los reject, major y minor y la tesis se iba animando poco a poco. El resultado son estas líneas escritas a finales de noviembre que marcan el comienzo del fin de esta tesis.

Este camino no lo he recorrido solo, encontrándome a gente en él que me han ayudado a llevar estos cuatro años que espero que me hayan ayudado a madurar a afrontar los diversos retos que me voy a encontrar cuando termine esta tesis. En primer lugar, dar las gracias a Paco y a Dani por darme la oportunidad de poder realizar esta tesis doctoral y haber confiado en mi y por las discusiones que hemos tenido a lo largo de estos años, que me han ayudado a crecer como persona y me han enriquecido como trabajador - o eso espero. También agradecer a Javi Del Ser por su contribución también en el desarrollo de la tesis, así como a Aritz y a Aitor por participar también en algunos estudios realizados en esta tesis, compartir con vosotros esos momentos también me lo llevo en mi mochila.

No podían faltar en estos agradecimientos a todos mis compañeros de la carrera por todo el tiempo que hemos pasado juntos. Obviamente a aquellos que hemos seguido en Granada como son Nuria, Juanlu, Elena, Paco e Iván por los buenos momentos y por echarme una mano cuando lo he necesitado, así como al resto de compañeros de despacho como Germán, Guille, José Daniel y, en particular a Nacho. La cantidad de paciencia que ha tenido conmigo para los experimentos en las gráficas es igual o mayor a la saturación que tenían estos cuando ponía todos mis procesos a correr.

Esta tesis no habría podido hacerlo tampoco sin ser capaz de liberar mi mente durante la semana para no estar pensando siempre en ella. El baloncesto ha sido y es la mejor vía de escape y gracias a la gente del club en el que estoy he conocido a personas que me han ayudado a llevar mejor estos años, empezando por Damián y Alfredo, con los que además he compartido clase, y otros compañeros y amigos del equipo como Jorge, Carlos, Ricardo, Coque, etc. No hemos conseguido ganar la liga, pero nos llevamos unos buenos momentos como equipo.

Agradecer a mis padres por darme la educación desde pequeño y permitir estudiar y desarrollarme en las diferentes etapas de mi vida, sin vosotros nada de esto sería posible y es algo de lo que os estaré eternamente agradecido. Gracias también a mi familia, a mi familia de Granada por acoger al niño que llegó en 2012 y ayudarme en todo lo necesario. También a mi familia de Sabiote porque también habéis velado por mi y siempre habéis

estado cuando os he necesitado. No me puedo olvidar tampoco de mi familia de Barcelona porque, aunque estéis más lejos, pero también habéis sido importantes y os he tenido en mi cabeza estos años. De igual manera, me quiero acordar también de mi abuela que, pese a que todo el tema de la informática quizás te quede lejos, pero para mi siempre estás cerca y espero disfrutar de tu presencia mucho más tiempo. Por último, he de acordarme también de mi abuelo Pepe que nos dejó justo antes de la pandemia, pero que la vida me ha permitido disfrutar de tu presencia y tu sabiduría durante 25 años. Espero que estés orgulloso de mi estés donde estés. Por último, a la persona con la que llevo compartiendo 5 años de mi vida, Vero, que ha sido mi pilar fundamental estos años. Cada día haces que sea mejor persona y gracias por todo el apoyo que me has dado, que ninguna línea de gradecimiento, por más que escriba, puede hacer que deje de agradecerte todo lo que has hecho y haces por mi.

Gracias.

Abstract

Artificial Intelligence (AI) is catalyzing a profound revolution across diverse industry sectors, reshaping the landscape of innovation and productivity with the development of general-purpose AI systems (GPAIS). Machine Learning (ML) is the field of AI that focuses on the study and development of algorithms that enable computers to perform tasks effectively by learning from data and improving through experience. The increasing impact of AI is notably evident in its ability to not only design novel systems but to enhance the efficiency of existing ones. ML models have greatly benefited from this revolution, as reflected in their optimization.

This AI revolution has been driven by the explosion of Deep Learning (DL), which uses neural networks to learn complex patterns from data. These models have been applied to solve a wide range of problems, having a great impact on society. The constant evolution of AI and the application of new concepts to ML subfields, such as DL, has only increased its importance in recent years. In addition, the existence of AI models capable of both designing and improving other AI models facilitates the realization of new approaches.

In this context, Metaheuristics (MHs) are optimization algorithms used to efficiently solve complex optimization problems, specifically when exact optimization methods become impractical due to the large search space or computational complexity of the problem. One of the families in the field of MH is the bio-inspired algorithms, which are inspired by the simulation of biological processes to create these optimization algorithms.

Within bio-inspired algorithms, Evolutionary Algorithms (EAs) constitute one of the most widely used algorithms for the design and optimization of models with desirable characteristics such as robustness and reliability. The extended trajectory of EAs in the optimization of ML models provides the exploration of new mechanisms that can be applied for both the design and enhancement of these models. This thesis presents as hypothesis the customization for the design of DL models using EAs since the EAs are capable of better adapting to the problem, improving the performance, while at the same time fostering other desirable properties such as robustness, diversity, and explainability.

This thesis addresses the following objectives:

1. The first objective involves a study of the field of MH and, specifically, the spectrum of bio-inspired algorithms in the literature. This study provides a comprehensive taxonomy encompassing all categories of bio-inspired algorithms, integrated with a dual analysis of the biological inspiration and the underlying mathematical model. The purpose of this survey is to examine potential connections between bio-inspired concepts and their mathematical representations. Additionally, this text presents an analysis of the field's evolution and notable proposals. It provides insight into the ongoing evolution and includes some notes about future directions.

2. The second objective involves the development of an EA to design DL models with better performance, but with fewer active neurons to overcome standard pruning methods. These methods focus on the reduction of the model but at the cost of worse results. The objective is to use EAs to design improved pruned networks by removing unnecessary neurons in a compatible way to import previous knowledge. Additionally, in this study, we conduct experiments to check whether the results are not caused by the EA's randomness and to evaluate the models' adaptation when new data are introduced.

3. The third objective consists of an extension of the previous study towards the design and optimization of DL models taking into account three objectives: performance, complexity, and robustness. For the sake of more interpretable DL models, an exploration of the most influential neurons and their representation in the original image is also performed. An ensemble strategy is used to enhance the performance and robustness of the DL model, by leveraging on the diversity of the initial models.

4. The fourth objective consists of an analysis of the role that Evolutionary Computation can play in the domain of GPAIS. The purpose of this work is to study the ability of EAs to design and enhance GPAIS. Moreover, this text presents how EA-based areas can be applied to fulfill the desired GPAIS properties. It also includes several examples of EAs to improve GPAIS. Lastly, it outlines the challenges of using EAs in GPAIS and the possible EA-based strategies to design or enhance GPAIS.

The objectives outlined in the thesis are successfully addressed. The objective related to the literature review contributes to the innovation of the research field, opening up several research directions for the usage of these algorithms toward different scopes in AI. The following two objectives, related to the creation of EA-based models for the design and enhancement of DL models, are supported by comparative empirical studies. Building upon the premise of removing unnecessary neurons, we have successfully designed networks with improved performance and robustness, while simultaneously reducing complexity. Finally, we present the work that provides an analysis of Evolutionary Computation, particularly about EAs, within the domain of GPAIS. Specifically, this analysis focuses

on the capacity of EAs to design and enhance these systems. Moreover, we align several research areas where EAs have gained a great influence to fulfill GPAIS properties and illustrate this synergy with several milestones. Also, we discuss the benefits of EAs for GPAIS and strategies based on EAs for the design and enhancement of GPAIS.

Resumen

La Inteligencia Artificial (*Artificial Intelligence* - AI) está llevando a cabo una profunda revolución en diversos sectores industriales, remodelando el panorama de la innovación y la productividad con el desarrollo de sistemas de AI de propósito general (*General-Purpose AI Systems* - GPAIS). El Aprendizaje Automático (*Machine Learning* - ML) es el campo de la AI que se centra en el estudio y desarrollo de algoritmos que permiten a los ordenadores realizar tareas eficazmente aprendiendo de los datos y mejorando a través de la experiencia. La creciente repercusión de la AI se hace patente, no solamente en su capacidad para diseñar sistemas novedosos, sino también para mejorar la eficacia de los ya existentes. Los modelos de ML se han beneficiado enormemente de esta revolución gracias a su optimización.

Esta revolución de la AI se ha visto impulsada por la explosión del Aprendizaje Profundo (*Deep Learning* - DL), que utiliza redes neuronales para aprender patrones complejos a partir de datos. Estos modelos se han aplicado para resolver una amplia gama de problemas, teniendo un gran impacto en la sociedad. La constante evolución de la AI y la aplicación de nuevos conceptos a subcampos del ML, como el DL, no ha hecho más que aumentar su importancia en los últimos años. Además, la existencia de modelos de AI capaces tanto de diseñar como de mejorar otros modelos de AI facilita la realización de nuevos enfoques.

En este contexto, las metaheurísticas (*Metaheuristics* - MHs) son algoritmos de optimización que se utilizan para resolver de forma eficiente problemas complejos de optimización, concretamente cuando los métodos de optimización exactos resultan poco prácticos debido al gran espacio de búsqueda o a la complejidad computacional del problema. Una de las familias del campo de las MH son los algoritmos bioinspirados, que se basan en la simulación de procesos biológicos para crear estos algoritmos de optimización.

Dentro de los algoritmos bioinspirados, los Algoritmos Evolutivos (*Evolutionary Algorithms* - EAs) constituyen uno de los algoritmos más utilizados para el diseño y optimización de modelos con características deseables como robustez y fiabilidad. La trayectoria extendida de los EAs en la optimización de modelos de ML proporciona la exploración de nuevos mecanismos que pueden ser aplicados tanto para el diseño como para la mejora de estos modelos. Esta tesis presenta como hipótesis la adaptación en el

diseño de modelos de DL utilizando EAs, puesto que los EAs son capaces de adaptarse mejor al problema, mejorando el rendimiento, a la vez que fomentan otras propiedades deseables como la robustez, la diversidad y la explicabilidad.

Esta tesis aborda los siguientes objetivos:

1. El primer objetivo consiste en un estudio del campo de las MH y, en concreto, del conjunto de algoritmos bioinspirados existentes en la literatura. Este estudio proporciona una taxonomía completa que abarca todas las categorías de algoritmos bioinspirados, integrada con un análisis dual tanto de la inspiración biológica como del modelo matemático subyacente. El propósito de este estudio es examinar las posibles conexiones entre los conceptos bioinspirados y sus representaciones matemáticas. Además, este texto presenta un análisis de la evolución del campo y propuestas notables. Se ofrece una visión de la evolución del campo a lo largo de estos últimos años e incluye algunas anotaciones sobre líneas de trabajo futuro.

2. El segundo objetivo implica el desarrollo de un EA para diseñar modelos de DL con mejores resultados, utilizando menos neuronas activas para superar los métodos de poda estándar. Estos métodos se centran en la reducción del modelo a costa de ofrecer peores resultados. El objetivo es utilizar los EAs para diseñar redes podadas mejoradas eliminando neuronas innecesarias de forma compatible para importar conocimiento previo. Además, en este estudio realizamos experimentos para confirmar que los resultados no se deben a la aleatoriedad del EA y también para evaluar la adaptación de los modelos conforme se introducen nuevos datos.

3. El tercer objetivo consiste en una extensión del estudio anterior hacia el diseño y optimización de modelos de DL teniendo en cuenta tres objetivos: rendimiento, complejidad y robustez. Además, con el objetivo de conseguir modelos de DL más interpretables, también se realiza una exploración de las neuronas más influyentes y su representación en la imagen original. Se utiliza una estrategia de *ensemble* para mejorar el rendimiento y la robustez del modelo DL, aprovechando la diversidad de los modelos iniciales.

4. El cuarto objetivo consiste en un análisis sobre el papel que la Computación Evolutiva puede jugar en el dominio de GPAIS. El propósito de este trabajo es estudiar la capacidad de los EAs para el diseño y mejora de los GPAIS. Además, este texto presenta cómo ciertas áreas basadas en EAs se pueden aplicar para satisfacer las propiedades deseadas de los GPAIS. También se incluyen varios ejemplos de EAs que mejoran GPAIS. Por último, se esbozan los retos que plantea el uso de los EAs en GPAIS y las posibles estrategias basadas en los EAs para diseñar o mejorar los GPAIS.

La tesis aborda los diferentes objetivos descritos de manera exitosa. El objetivo relacionado con el estudio de la literatura aporta innovación al campo de investigación, mejorando

la literatura ya existente al mismo tiempo que abre diferentes líneas de investigación futuras en diferentes ámbitos de la AI. Los siguientes dos objetivos, relacionados con la creación de modelos basados en los EAs para el diseño y mejora de modelos de DL, están respaldados por estudios empíricos comparativos. Partiendo de la premisa de eliminar las neuronas innecesarias, hemos diseñado con éxito redes mejoradas en rendimiento y robustez, reduciendo al mismo tiempo la complejidad. Por último, presentamos el trabajo que proporciona el análisis de la Computación Evolutiva, en particular sobre los EAs, dentro del dominio de los GPAIS. En concreto, se centra en la capacidad de los EAs para diseñar y mejorar estos sistemas. Además, alineamos varias áreas de investigación en las que los EAs han adquirido una gran influencia para las propiedades de los GPAIS e ilustramos esta sinergia con varios hitos. Asimismo, proponemos los beneficios de los EAs para los GPAIS y las estrategias basadas en estos algoritmos para el diseño y la mejora de los GPAIS.

Table of Contents

List of Abbreviations

AI Artificial Intelligence

CKA Centered Kernel Alignment

DL Deep Learning

EAs Evolutionary Algorithms

EDL Evolutionary Deep Learning

GPAIS General-Purpose AI Systems

MHs Metaheuristics

MOEAs Multi-Objective Evolutionary Algorithms

ML Machine Learning

NSGA-II Nondominated Sorting Genetic Algorithm II

OoD Out-of-Distribution Detection

PF Parento Front

TL Transfer Learning

XAI eXplainable Artificial Intelligence

Chapter I

PhD Dissertation

1 Context

Evolutionary Algorithms (EAs) [BFM97] represent a class of Metaheuristics (MHs) that have significantly impacted the field of computational intelligence [Sim13]. Widely applicable in real-world scenarios [CWM12], EAs integrate principles from biology with optimization strategies. They emulate natural selection, genetic variation, and survival of the fittest to iteratively improve a population of candidate solutions. Although EAs have a stochastic nature, and they do not guarantee to achieve the optimum value, they present competitive solutions for optimization problems with limited resources.

Their success has produced an overwhelming growth of these algorithms, reflected in taxonomies [SEBB20], but has also led to criticisms related to the novelty and application benefits of recent proposals [CVSD20, TD21]. However, it is undeniably their applicability in complex optimization problems [Yan10]. From this practical perspective, a very important advantage is the ability to optimize several objectives at the same time, as in the case of the Multi-Objective Evolutionary Algorithms (MOEAs) [CLVV$^+$14]. EAs offer a framework for tackling diverse optimization challenges, leveraging collective intelligence to foster innovation and problem-solving [DSOM$^+$19].

Machine Learning (ML) is a subfield of the Artificial Intelligence (AI) which aims to develop models that learn about data and extract patterns from it. Through a longstanding history of collaboration, EAs are the bio-inspired algorithms that have contributed most to both the design and optimization of ML models, driving advancements in the field [ASBC$^+$19, TTBG21, LMX$^+$23]. This collaborative approach, known as *AI-powered AI*, involves employing one AI (such as EAs) to design or enhance another AI model. The concept has garnered considerable attention and research within the ML community [STÖ19].

Deep Learning (DL) [GBC16] is one of the branches of AI that has led the technological revolution of the last years. DL is a dynamic field of research, with continuous advancements in model architectures, optimization algorithms, and interpretability techniques that expand the limits of AI. These models learn to automatically discover relevant patterns and relationships from large datasets and offer a qualitative improvement to other existing techniques, as in the case of image classification [RW17]. Despite its immense practical applications, designing a suitable neural network for the problem to be solved is a complicated task. For this reason, it is common to use existing architectures for multiple problems.

There are two relevant aspects about DL that we want to highlight due to their importance in this thesis. The first one is the Transfer Learning (TL) [PY09]. It involves using the knowledge gained from solving one or more problems to solve a different but related problem. As a result of using a pretraining model, the knowledge of the previous problem can be used for the new one, enabling efficient learning and maintaining model quality with fewer examples and computational resources. Another relevant feature that we are going to consider is the robustness of the DL model. To measure it, frameworks for robustness have been developed such as the Out-of-Distribution Detection (OoD), where the ability of the model to handle the unknown is tested, specifically, to detect

whether it has been queried with an example of a not learned distribution, as with *Out-of-DIstribution detector for Neural networks* (ODIN) [LLS18].

Since EAs have been used for the design and optimization of ML models, their application can bring benefits to the field of DL. As a consequence, the increasing application of EAs in DL has raised the field of Evolutionary Deep Learning (EDL). This field is an alternative to learn weights and hyper-parameters for the DL models, but also for the design of the configuration of the layers. Figure 1 shows a scheme of a standard EDL process. These approaches have been of interest to researchers and hundreds of works have been developed to design or optimize DL models, as in [ALMR19]. Their importance has escalated to the point of developing surveys and taxonomies aimed to guide the design and optimization of these models [MDVR$^+$21, LMY$^+$23].

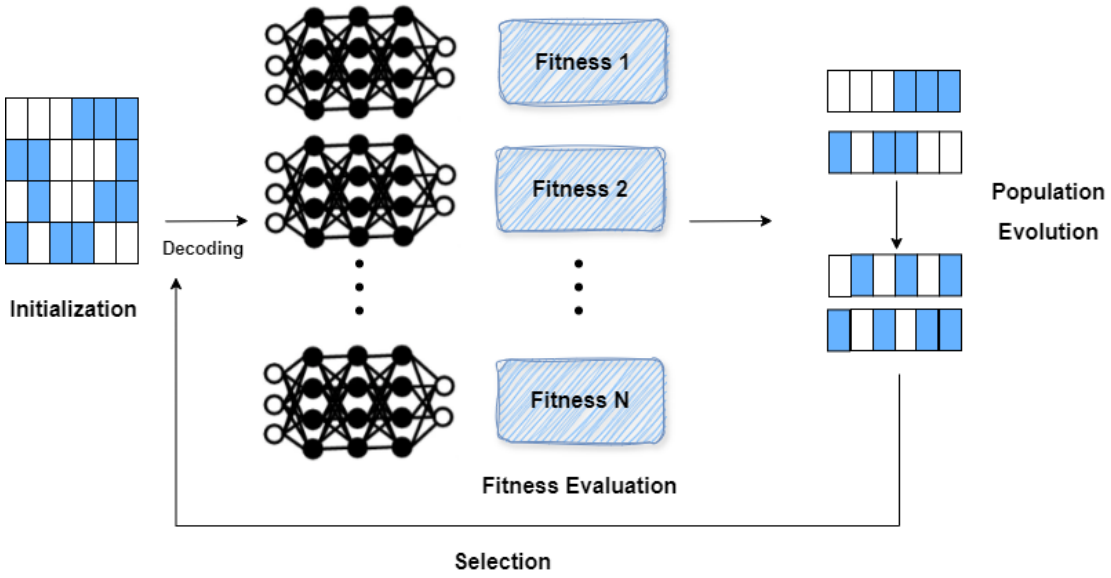


Figure 1: Schematic representation of an EDL process.

While EAs have historically been associated with optimization tasks, there is growing evidence to suggest that their design capabilities are equally formidable. Within EDL, there is a branch called Neural Architecture Search (NAS), which addresses the problem of finding and adapting the network architecture through a search process. An evolutionary strategy can be used to guide the search that finds the best configuration of layers or even the backbone of the network. The majority of developments in these last years have driven detailed studies of certain aspects of DL related to NAS, such as automatic network design [ZQGT21, ÜB22, LSX$^+$23].

This thesis considers as hypothesis, grounded in the background of EDL, that the integration of EAs could design enhanced DL models. Consequently, this thesis aims to design DL models with superior performance by using EAs. Specifically, we intend to utilize TL and OoD paradigms to develop networks with enhanced performance metrics,

while simultaneously preserving lower complexity and other desirable attributes such as robustness.

The integration of EAs with DL falls within the broader domain of General-Purpose AI Systems (GPAIS). Various definitions of GPAIS have been proposed [GAU+23, CL23], with Triguero et al. offering a comprehensive analysis and a dual definition based on their adaptability to new tasks [TMP+24]. They also introduce a taxonomy grounded in the concept of *AI-powered AI* for system design and optimization, which is pertinent to our use of EAs in designing and enhancing DL models. Many EDL proposals are aligned with the notion of *closed-world GPAIS* given in [TMP+24], exemplified by notable works such as [RLSL20] or [MLM+24b]. On the other hand, the *open-world GPAIS* are related to works in which prior knowledge is used to adapt to new tasks in innovative ways, such as the evolutionary generation of new environments with POET [WLCS19], the generation of diverse models for learning in zero-shot environments [CCH+23]. These examples are some of the first EA-based approaches that improve these systems, suggesting the potential of EAs for GPAIS.

Contexto

Los Algoritmos Evolutivos (*Evolutionary Algorithms* - EAs) [BFM97] representan una clase de Metaheurísticas (*Metaheuristics* - MHs) que han tenido un impacto significativo en el campo de la inteligencia computacional. Ampliamente aplicables en escenarios del mundo real [CWM12], los EAs integran principios de la biología con estrategias de optimización. Emulan la selección natural, la variación genética y la supervivencia del más apto para mejorar iterativamente una población de soluciones candidatas. Aunque los EAs tienen una naturaleza estocástica y no garantizan alcanzar el valor óptimo, presentan soluciones competitivas para problemas de optimización con recursos limitados.

Su éxito ha producido un crecimiento desmesurado de estos algoritmos que se refleja en las taxonomías [SEBB20], pero también ha dado lugar a críticas relacionadas con la novedad y las ventajas a la hora de la aplicación de las propuestas recientes [CVSD20, TD21]. Sin embargo, es innegable su aplicabilidad en problemas complejos de optimización [Yan10]. Desde esta perspectiva práctica, una ventaja muy importante es la capacidad de optimizar varios objetivos al mismo tiempo, como en el caso de los Algoritmos Evolutivos Multi-Objetivo (Multi-Objective Evolutionary Algorithms - MOEAs) [CLVV$^+$14]. Los EAs ofrecen un marco de trabajo para abordar diversos retos de optimización, aprovechando la inteligencia colectiva para fomentar la innovación y la resolución de problemas [DSOM$^+$19].

El Aprendizaje Automático (*Machine Learning* - ML) es un subcampo de la Inteligencia Artificial (*Artificial Intelligence* - AI) cuyo objetivo es desarrollar modelos que aprendan sobre los datos y extraigan patrones de ellos. A través de una larga historia de colaboración, los algoritmos bioinspirados son los que más han contribuido tanto al diseño como a la optimización de los modelos de ML, impulsando los avances en este campo [ASBC$^+$19, TTBG21, LMX$^+$23]. Este enfoque colaborativo, conocido como IA que potencia a otra IA (*AI-powered AI*), implica el uso de una AI (como EAs) para diseñar o mejorar otro modelo de AI. Este concepto ha suscitado una atención y una labor de investigación considerables en el seno de la comunidad del ML [STÖ19].

El Aprendizaje Profundo (*Deep Learning* - DL) [GBC16] es una de las ramas de la AI que ha liderado la revolución tecnológica de los últimos años. El DL es un campo de investigación dinámico, con continuos avances en arquitecturas de modelos, algoritmos de optimización y técnicas de interpretabilidad que amplían los límites de la AI. Estos modelos aprenden a descubrir automáticamente patrones y relaciones de relevancia a partir de grandes conjuntos de datos y ofrecen una mejora cualitativa a otras técnicas existentes, como en el caso de la clasificación de imágenes [RW17]. A pesar de sus inmensas aplicaciones prácticas, diseñar una red neuronal adecuada para el problema a resolver es una tarea complicada. Por este motivo, es habitual utilizar arquitecturas ya existentes para múltiples problemas.

Hay dos aspectos significativos sobre el DL que queremos destacar por su importancia en esta tesis. El primero es el Aprendizaje por Transferencia (*Transfer Learning* - TL)

[PY09]. Consiste en utilizar los conocimientos adquiridos en la resolución de uno o varios problemas previos para resolver un problema diferente, pero relacionado. Como resultado del uso de un modelo preentrenado, el conocimiento del problema anterior puede ser utilizado para el nuevo, permitiendo un aprendizaje eficiente y manteniendo la calidad del modelo con menos ejemplos y recursos computacionales. Otra característica relevante que vamos a considerar es la robustez del modelo de DL. Para medirla, se han desarrollado marcos de trabajo para medir la robustez como la Detección fuera de la Distribución (Out-of-Distribution Detection - OoD), donde se prueba la capacidad del modelo para manejar la incógnita, en concreto, para detectar si dicho modelo ha sido consultado con un ejemplo de una distribución no aprendida, como ocurre con *Out-of-DIstribution detector for Neural networks* (ODIN) [LLS18].

Dado que los EAs se han utilizado para el diseño y optimización de modelos ML, su aplicación puede aportar beneficios al campo del DL. Como consecuencia, la creciente aplicación de los EAs en el DL ha propiciado el nacimiento del campo del Aprendizaje Profundo Evolutivo (*Evolutionary Deep Learning* - EDL). Este campo es una alternativa para aprender pesos e hiperparámetros para los modelos de DL, pero también para el diseño de la configuración de las capas. La Figura 2 muestra un esquema de un proceso de EDL estándar. Estas aproximaciones han sido del interés de los investigadores y se han desarrollado cientos de trabajos para diseñar u optimizar modelos de DL, como en [ALMR19]. Su importancia se ha incrementado hasta el punto de desarrollar estudios y taxonomías orientadas a guiar el diseño y optimización de estos modelos [MDVR+21, LMY+23].

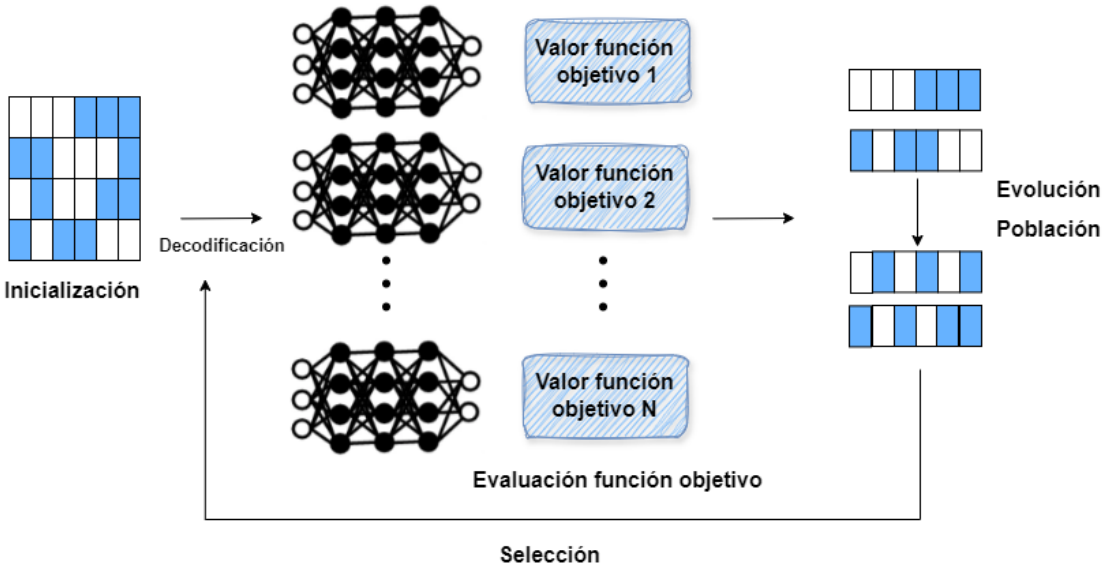


Figura 2: Representación esquemática de un proceso en EDL.

Aunque los EAs se han asociado históricamente con tareas de optimización, cada vez hay más indicios que sugieren que sus capacidades de diseño son también formidables.

Dentro del EDL, existe una rama denominada Búsqueda de Arquitectura Neuronal (Neural Architecture Search - NAS), que aborda el problema de encontrar y adaptar la arquitectura de red mediante un proceso de búsqueda. Se puede utilizar una estrategia evolutiva para dicha búsqueda, que encuentra la mejor configuración de capas o incluso la columna vertebral de la red. La mayoría de los desarrollos de estos últimos años han propiciado estudios detallados de ciertos aspectos del DL relacionado con NAS, como el diseño automático de redes [ZQGT21, ÜB22, LSX+23].

Esta tesis considera como hipótesis, basándose en los antecedentes del EDL, que la integración de los EAs podría permitir diseñar modelos DL mejorados. En consecuencia, el objetivo de esta tesis es diseñar modelos DL con prestaciones superiores mediante el uso de los EAs. En concreto, pretendemos utilizar los paradigmas del TL y del OoD para desarrollar redes con métricas de rendimiento mejoradas, preservando simultáneamente una menor complejidad y otros atributos deseables como la robustez.

La integración de los EAs con el DL se enmarca dentro de un ámbito más general, los sistemas de AI de propósito general (*General-Purpose Artificial Intelligence Systems* - GPAIS). Se han propuesto varias definiciones de los GPAIS [GAU+23, CL23], pero en Triguero et al. se ofrece un análisis exhaustivo y una definición dual basada en su adaptabilidad a nuevas tareas [TMP+24]. También se introduce una taxonomía basada en el concepto de *AI-powered AI* para el diseño y optimización de sistemas, que es coherente con nuestro uso de los EAs en el diseño y mejora de modelos de DL. Muchas propuestas del EDL están alineadas con la noción de GPAIS de mundo cerrado (*closed-world GPAIS*) dada en [TMP+24], ejemplificada por trabajos notables como [RLSL20] o [MLM+24b]. Por otro lado, los GPAIS de mundo abierto (*open-world GPAIS*) están relacionados con trabajos en los que se utiliza el conocimiento previo para adaptarse a nuevas tareas de forma innovadora, como la generación evolutiva de nuevos entornos con POET [WLCS19] o la generación de diversos modelos para el aprendizaje en entornos de *few-shot learning* [CCH+23]. Estos son unos de los primeros modelos basados en EAs que permiten la mejora de estos sistemas, sugiriendo el potencial de los EAs para los GPAIS.

2 Hypothesis

In the field of ML, EAs have been established as powerful tools for designing and optimizing models. Despite some drawbacks, such as lack of global optimization guarantees, and slow convergence rates, they possess several characteristics that make them a compelling option for solving various problems. These characteristics include easy implementation, adaptability which allows for dynamic adjustment of search strategies based on problem landscapes, and versatility in customizing operators to adapt to different problem domains.

EDL has gained significant relevance due to its ability to utilize evolutionary strategies for optimizing DL models. Additionally, it encompasses various learning paradigms adaptable to diverse data types and scenarios based on their complexity. Notably, EDL methods have evolved alongside the rise of DL and have shown promise in tackling challenging problems. The specific motivations underlying this thesis are outlined below.

- Bio-inspired algorithms have been applied in various domains. Surveys have classified these algorithms based on their natural or biological inspiration, resulting in numerous categories. However, the mathematical model of the bio-inspired algorithm is often overlooked, despite being the most crucial characteristic of the algorithm. The hypothesis is that the diversity of the mathematical model is lower than the diversity offered by the biological inspiration. The analysis and classification from the perspective of a taxonomy could reflect this assumption.

- Many EDL proposals focus on adapting the convolutional phase of the DL model. We hypothesize that we can design DL models with better performance by adapting the last layers and eliminating unnecessary connections. By doing so, we can also leverage previous knowledge from similar problems to enhance the efficiency of resource utilization.

- MOEAs are known for their robustness and ability to handle diverse scenarios effectively, often yielding interpretable solutions compared to black-box optimization methods. For these reasons, our hypothesis is that the utilization of OoD as another objective of the MOEA can facilitate the design of more interpretable and robust DL models.

- Recent trends suggest that GPAIS show the need to move towards more open and robust approaches. Our hypothesis considers that EAs could be relevant in the scope of GPAIS. The adaptability and robustness of EAs may play a crucial role in the design and enhancement of GPAIS.

To summarize, the proposed thesis project constitutes a substantiated contribution to the advancement of EDL. The impact of this thesis is justified by the development of a comprehensive survey of bio-inspired algorithms based on the mathematical model which serves as a reference for an analysis of the field, two proposals based on EAs for

the design of improved, more robust and more explicable DL models, and a study about the potential of EAs for the design and enhancement of GPAIS.

3 Objectives

After establishing the hypothesis of this thesis, the following section elaborates on the objectives that have driven it. **The main objective is to use evolutionary strategies within the scope of DL. Specifically, the objective of this thesis is to develop EAs for designing improved DL models for image classification in terms of performance while prioritizing properties like robustness and explainability.**

To accomplish this, a comprehensive survey on MHs was created, which presents a convenient taxonomy to classify bio-inspired algorithms like EAs in terms of natural inspiration and mathematical model. Then, two more objectives of this thesis are related to the design and optimization of DL models using EAs. Lastly, the potential of EAs in GPAIS is also studied. These objectives can be broken down as follows:

Study of the bio-inspired algorithms in the MHs field, resulting in a comprehensive survey and taxonomy. To accomplish our first objective, we conduct an extensive literature review aimed at establishing a comprehensive taxonomy grounded in the similarities and distinctions among bio-inspired optimization algorithms. Through this research, we categorize all MHs inspired by biological processes and nature into distinct categories based on their biological and mathematical characteristics, to determine whether the diversity offered by mathematical model-based inspiration is similar to that offered by biological inspiration. We examine the similarities between novel and classical bio-inspired algorithms and evaluate the strengths, weaknesses, and recent advancements in the field. We aim to address the limitations of the field by reviewing several studies, guidelines, overviews, taxonomies, and general approaches to serve as a guide for other researchers. We highlight the potential research lines of the field.

To develop an evolutionary pruning model to automatically design improved DL models. The specific design of DL models represents a challenging task. Many existing evolutionary strategies in the literature demand substantial computational resources to formulate entire models. Hence, the goal is to develop an EA tailored for designing the fully-connected layers of DL models. This approach leverages TL to import and apply knowledge from related problems, followed by pruning to eliminate redundant neurons within these layers. The objective is to design networks with improved performance while minimizing, at the same time, complexity. We will study the robustness of the designed models using the Centered Kernel Alignment (CKA). To evaluate the effectiveness of our approach, we will implement and compare it against several prominent pruning methods for fully-connected layers.

To develop a multi-objective evolutionary pruning model to automatically design improved DL models. Many approaches in DL model design using a MOEA primarily emphasize optimizing performance and complexity. However, they often overlook the crucial aspect of robustness. This study aims to create a specialized MOEA for DL model design that considers performance, complexity, and robustness as concurrent

objectives. In addition, we will utilize the GradCam, an eXplainable Artificial Intelligence (XAI) technique to elucidate the regions of original images associated with influential neurons. We will use an ensemble modeling strategy to capitalize on model diversity to further improve the performance and robustness of the designed DL models. Lastly, we will conduct comparative analyses against existing pruning methods to evaluate the efficacy of our proposed approach.

Study about the role of Evolutionary Computation in the GPAIS field, resulting in a position paper about the potential of EAs in GPAIS. In a position paper, we analyze the possible role of Evolutionary Computation, specifically EAs, in the field of GPAIS. We study how EAs can design or improve GPAIS. We describe the way in which EA-based areas can be applied to fulfill the desired GPAIS properties. The analysis shows recent milestones on EAs and GPAIS. To discover and encourage future research directions, we discuss the challenges in exploiting the advantages of using EAs in GPAIS. Also, we present strategies that can be implemented with EAs to both design and improve GPAIS, along with EA-based research areas that can help to realize them. This work can be useful to deal with the recent challenges in the AI about the design and enhancement of GPAIS.

4 Methodology

The research conducted throughout this thesis has been carried out following the scientific method. In this particular case, it requires both practical and theoretical methodologies. The general guidelines applied in all studies included in this thesis are summarized here:

- **Observation**: through the study of the EAs task, and focusing on DL. The goal of this stage is to identify research opportunities, which could result in new, successful models to address EAs in the realm of DL and extend its applicability.
- **Formulation of hypotheses**: design of new EAs algorithms for DL, with an emphasis on their scalability with respect to the amount of constraint-based information available. The models designed and developed must fulfill the objectives described in previous sections.
- **Experimental data collection**: the designed models are tested on diverse scenarios to obtain results as representative of their capabilities as possible. These results are later analyzed using external quality indices.
- **Contrasting the hypotheses**: the results obtained are compared with representative models from the existing literature, to analyze their quality in terms of efficiency and effectiveness. To this end, a set of representative models is chosen on the basis of a comprehensive literature review. These models are implemented and published, for the sake of reproducibility of results.
- **Validation of hypotheses**: hypotheses formulated in the experiments are proven or disproven following objective quality indicators and statistical testing. If any given hypothesis is rejected, it must be modified and the previous steps repeated from that point on.
- **Scientific thesis**: relevant conclusions are extracted in view of the outcomes of the research process. All the results and conclusions obtained must be gathered and synthesized into a documentary report of the thesis.

5 Thesis Developments

The body of knowledge compiled in this thesis is found in 4 different studies, three of them published in scientific journals. This section aims to introduce and summarize both the purpose and the developments of these studies, whose results will be discussed later (in Section 6). The publications are listed below:

- Molina, D., Poyatos, J., Del Ser, J., García, S., Hussain, A., & Herrera, F. (2020). Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. Cognitive Computation, 12, 897-939. DOI: `https://doi.org/10.1007/s12559-020-09730-8`
- Poyatos, J., Molina, D., Martinez, A. D., Del Ser, J., & Herrera, F. (2023). EvoPruneDeepTL: An evolutionary pruning model for transfer learning based deep neural networks. Neural Networks, 158, 59-82. DOI: `https://doi.org/10.1016/j.neunet.2022.10.011`
- Poyatos, J., Molina, D., Martínez-Seras, A., Del Ser, J., & Herrera, F. (2023). Multiobjective evolutionary pruning of Deep Neural Networks with Transfer Learning for improving their performance and robustness. Applied Soft Computing, 147, 110757. DOI: `https://doi.org/10.1016/j.asoc.2023.110757`.

The rest of this section is organized according to the publications listed above, and the objectives described in Section 3. Firstly, Section 5.1 presents a survey on MHs, including a dual taxonomy based on the biological inspiration and mathematical model, with an analysis of the most influential algorithms, an evolution of the field in recent years, and the future challenges of the research area. In Section 5.2, we focus on EAs and DL to propose a new EAs to design DL models for image classification to improve the performance. In Section 5.3, we extend the previous evolutionary model using a MOEA to design and enhance DL models using three objectives. Furthermore, in both cases, we carry out several experiments to analyze the diversity and robustness of the models. Lastly, in Section 5.4, we describe the analysis made in the position paper about the potential of EAs in GPAIS. In Chapter III, this work is deeply described, both in content and structure.

5.1 Study of the bio-inspired algorithms in the MH field, resulting in a comprehensive survey and taxonomy

The field of MHs has undergone extensive study over the past few decades, resulting in a plethora of algorithms that have demonstrated exceptional performance across various domains. Despite this, existing surveys and taxonomies often focus primarily on the biological inspiration behind these algorithms, resulting in repetitive classifications

with limited algorithmic diversity and minimal advancement. Consequently, while both bio-inspired algorithms and MHs techniques are extensively studied, there remains a significant gap in the literature for a comprehensive survey that encompasses all possible bio-inspired algorithms with a taxonomy based on their underlying mathematical models.

This survey comprehensively covers the field of bio-inspired algorithms, beginning with an overview of the concept and previous surveys in the field. We then introduce two proposed taxonomies: one based on biological inspiration and the other on the underlying mathematical models. Each taxonomy is explored in detail, categorizing algorithms based on their inspiration and solution-creation methods. Additionally, we analyze how classical algorithms have influenced newer ones. The study includes a critical analysis of bio-inspired algorithms, evaluating their strengths, weaknesses, and areas for improvement. Notable papers from recent years, offering guidelines for researchers and future directions, other taxonomies, overviews, and general approaches are also discussed. Finally, we draw conclusions based on the insights gained from analyzing more than 500 algorithms.

The publication associated with this study is:

> Molina, D., Poyatos, J., Del Ser, J., García, S., Hussain, A., & Herrera, F. (2020). Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. Cognitive Computation, 12, 897-939. DOI: `https://doi.org/10.1007/s12559-020-09730-8`

5.2 To develop an evolutionary pruning model to automatically design improved DL models

The design of DL models is a complicated task due to the vast set of possibilities that may fit the problem at hand. Various automatic approaches exist for designing these networks, each with its advantages and drawbacks. One such approach is the pruning technique, which involves eliminating unnecessary connections in the network to create a more streamlined version. However, relying solely on pruning may lead to a decrease in performance. The use of TL can speed up the network design process by leveraging knowledge from previously studied problems. Therefore, there is a need to develop an EA capable of evolving pruning patterns for the fully-connected layers to design DL models with better performance.

In this work, we have developed an evolutionary pruning scheme that refines the TL process for designing improved DL models. The TL, when applied to the convolutional phase of the neural network with similar problems, allows the use of EAs in the fully-connected phase, where the model learns to distinguish between classes. Pruning is then applied to eliminate unnecessary neurons, resulting in a population of networks that EAs evolve towards a designed network with an optimized pruning pattern. This novel workflow breaks from traditional pruning methods by aiming to enhance model performance rather than degrade it. We also address two crucial aspects of DL: robustness

and adaptation to new data. The CKA measure provides insights into the robustness of the designed networks. Additionally, introducing a new class incrementally demonstrates the approach's adaptability to different scenarios.

The publication associated with this study is:

Poyatos, J., Molina, D., Martinez, A. D., Del Ser, J., & Herrera, F. (2023). EvoPruneDeepTL: An evolutionary pruning model for transfer learning based deep neural networks. Neural Networks, 158, 59-82. DOI: https://doi.org/10.1016/j.neunet.2022.10.011

5.3 To develop a multi-objective evolutionary pruning model to automatically design improved DL models

The field of EDL primarily focuses on optimizing DL models based on performance and complexity metrics. However, integrating robustness as an additional objective is a novel approach. Robustness, often measured through OoD detection, evaluates a model's ability to distinguish between known and unknown samples, essential for addressing open-world problems. To address this, there is a need to develop a MOEA that considers performance, complexity, and robustness, specifically targeting the evolution of pruning patterns in fully-connected layers.

In this study, we propose a novel approach based on a MOEA that refines the TL process to design improved DL models. The MOEA simultaneously enhances performance, and robustness, and reduces complexity by generating diverse models. Following a similar workflow to previous work, TL is applied to the convolutional phase while the MOEA, combined with pruning, operates in the fully-connected phase. This approach yields DL models exhibiting diversity across all objectives in the Parento Front (PF). Additionally, we conduct three sets of experiments: (1) comparison against pruning techniques, (2) analysis of influential neurons using a XAI technique, providing insight into class prediction regions in images, and (3) utilization of an ensemble to capitalize on the MOEA's ability to generate diverse models, aiming to improve performance, robustness, and mitigate overfitting risks.

The publication associated with this study is:

Poyatos, J., Molina, D., Martínez-Seras, A., Del Ser, J., & Herrera, F. (2023). Multiobjective evolutionary pruning of Deep Neural Networks with Transfer Learning for improving their performance and robustness. Applied Soft Computing, 147, 110757. DOI: https://doi.org/10.1016/j.asoc.2023.110757.

5.4 Study about the role of Evolutionary Computation in the GPAIS field, resulting in a position paper about the potential of EAs in GPAIS

In recent years, the optimization of ML models has been a widely studied problem. Among the various possibilities, EAs have achieved great importance in ML. The latest advances in AI indicate a shift towards the development of GPAIS, and their optimization constitutes a challenging task. The utilization of well-proven algorithms such as EAs in this new field is an opportunity to prove their value. Therefore, there is a need for a position paper that encompasses the role of EAs in GPAIS and how they design and enhance these systems and strategies to address advances in the realization of GPAIS using EAs, to foster more research in this field.

This work briefly explains the historical importance of Evolutionary Computation in ML, the field of GPAIS in terms of their definition and properties, and a taxonomy based on *AI-powered AI* to design and/or enhance GPAIS. By leveraging on this taxonomy but adapted for EAs (called *EA-powered AI*), these algorithms serve as an additional layer of abstraction. This position paper provides insights about how GPAIS can be designed and enhanced using EAs. These models are referred to as EA-GPAIS. To underscore the possible connection between GPAIS and EAs, their properties are matched to EA-based research areas. Several milestones in EA-GPAIS are shown as a result of the synergy between them in recent years. This study concludes with a description of the challenges to harness the benefits of using EAs in GPAIS and with several strategies that can be implemented using EAs to both design and improve GPAIS. This last analysis serves as motivating evidence for the present and future development of EA-GPAIS.

6 Discussion of Results

The first and last objectives of the thesis do not include experimental studies, but for the other objectives, a large experimental section has been developed to test the performance of each proposal. We have designed a homogeneous experimental setup that ensures both research conclusions and robustness in terms of the conclusions of the experiments so that this setup guarantees a fair comparison between different approaches. We follow the recommended guidelines through the design of all the experimental setups for the sake of consistency and validity of the findings among the wider academic community. The pruning algorithms that have been used in our comparison have been developed following their recommendations.

This section summarizes the discussion of the results obtained to achieve the objectives of this thesis and provides the analyses carried out based on the experimental results. Similarly to Section 5, the rest of this section is summarized according to the publications and the objectives introduced in Section 3. Section 6.1 provides the recommendations and conclusions from the study of the literature in MHs. Section 6.2 shows the results obtained in our first evolutionary pruning model (EvoPruneDeepTL). Section 6.3 displays the results of the MOEA for the pruning of DL models (MOEvoPruneDeepTL). Lastly, Section 6.4 presents the conclusions from the position paper about the potential of EAs and GPAIS.

6.1 Study of the bio-inspired algorithms in the MH field, resulting in a comprehensive survey and taxonomy

The study related to this objective provides an overview of bio-inspired optimization algorithms. It starts with an introduction about bio-inspired computation and MHs, analyzing their strengths and weaknesses. It follows with a review of recent surveys of bio-inspired algorithms and the dual taxonomy. The first taxonomy is based on biological inspiration, providing more specialized categories compared to previous surveys. The second taxonomy, a novel addition, classifies algorithms based on their underlying mathematical model. Additionally, the study analyzes influential algorithms from the literature and those closely derived from them. During this study, a total of 500 bio-inspired algorithms were reviewed.

The study and proposed taxonomies offer valuable insights:

- Provide a comprehensive basis for learning about bio-inspired optimization algorithms and the most promising research directions.
- Categorize all other bio-inspired proposals in each category according to their characteristics. Almost 25% of the reviewed algorithms are based on the classical MHs.

- Identify the most used bio-inspired optimization algorithms and provide several lessons for the reader to follow to design new bio-inspired algorithms.
- Provide an analysis of the situation of the field, its strengths, weaknesses, and future prospects. We show a summary of guidelines for complete design and perform a fair comparison between bio-inspired algorithms.

This study has provided several insights into the shortcomings of bio-inspired optimization, as well as its strengths and capabilities for developing innovative algorithms. This extensive review has contributed to the recognition of several promising research directions of the fields and lessons learned from the evolution of the bio-inspired optimization field. These directions are highlighted below:

- Behavior is more relevant than natural inspiration, as the existing literature is saturated with numerous nature- and bio-inspired algorithms. Our taxonomies show that algorithms from different sources of inspiration often exhibit similar behavior, which is a bad and unexpected result, since many — and of various categories — bio-inspired algorithms present such similarities when they come from totally different domains. This prevents the field from higher progress due to the high similarity between proposals.
- Nature-based terminology can make it difficult to understand the proposal because of the terminology used to describe it. Instead, to make proposals more understandable, it would be desirable for the description of the algorithm to be defined in an inspiration-agnostic way.
- Good comparisons are crucial for new proposals since most new proposals are compared to the same classical algorithms and not to state-of-the-art algorithms.
- A handful of guidelines for the design and comparison of new bio-inspired algorithms, so that research could easily develop several experiments appropriately to compare the new algorithm.
- More recent taxonomies, overviews, and general approaches of MHs are also presented to show the ongoing evolution of the field.
- A present and future abundance of exciting applications where bio-inspired algorithms and other approaches have been used in various domains to address real-world applications, as well as their application in ML and DL, which also gives insight into the importance.

6.2 To develop an evolutionary pruning model to automatically design improved DL models

Our proposal for the evolutionary pruning model which uses TL to import previous knowledge is called EvoPruneDeepTL. We have created a repository to offer the source

code for other researchers, following the recommendation of open science[1]. Although the same process could be applied to other domains, we have tested it on image classification problems. We decided to use several datasets of different domains, to check that our proposal behaves correctly. We have used a set of representative datasets in our experiments, including medical images, vegetal images, and paintings, among others.

We have developed a binary EAs that masks its chromosomes in pruned networks, and we have tested different networks and scenarios. We considered a standard fully-connected network with one and two layers and all their possibilities as baselines. We have also used several pruning methods from the literature with excellent results in pruning fully-connected layers to compare our proposal. For the evaluation, we use the standard evaluation metrics based on the performance in terms of accuracy. EvoPruneDeepTL achieves improved DL models in terms of performance and, although it is not its primary goal, reduces complexity by eliminating unnecessary neurons.

The study brings to the table the following valuable insights:

- The results indicate that pruned networks evolve to perform better than other pruning methods that assume lower performance. They outperform all considered baselines in all experimental settings.
- The best results are achieved when the removed neurons are taken directly from the learning acquired from the transfer learning, resulting in a feature selection approach. Additionally, good results are also achieved when the removed neurons are the connections between layers, which is even better than other pruning approaches.
- The CKA measure indicates that the results of our proposal are not due to the randomness of the EA, thus giving robustness as a consequence.
- The step-by-step incorporation of new data into the model demonstrates the adaptability of our proposal and improves the performance.

6.3 To develop a multi-objective evolutionary pruning model to automatically design improved DL models

Our work of multi-objective evolutionary pruning addresses the same problem as the image classification discussed before and the same datasets are used. We also follow the same recommendations and create a repository to free the code for other researchers [2]. Concerning the multi-objective algorithm, we use a modified version of Nondominated Sorting Genetic Algorithm II (NSGA-II) in which we have adapted the operators. In addition, we have implemented an ensemble to take advantage of the diversity of models generated with the MOEA. We have also used the GradCam to analyze the most influential

[1] `https://github.com/ari-dasci/S-EvoPruneDeepTL`
[2] `https://github.com/ari-dasci/S-MOEvoPruneDeepTL`

neurons and their representation in the original images. For the PF of the solutions, we took the best solutions of all runs of the algorithm. The OoD detection was done with the ODIN method.

The study offers the following valuable insights:

- The evolution of pruning patterns that eliminate unnecessary neurons enables the design of networks with improved accuracy, complexity, and robustness.
- Our proposal outperforms other competitive pruning methods based on the accuracy metric.
- The designed networks exhibit high values of accuracy and robustness, while the complexity is low.
- We have collected the most influential neurons of the PF and, using the GradCam, we have displayed the region of the original image associated with these neurons.
- Exploiting the diversity of DL models, the ensemble contributes to improving these results in terms of accuracy and robustness in all the scenarios studied.

6.4 Study about the role of Evolutionary Computation in the GPAIS field, resulting in a position paper about the potential of EAs in GPAIS

This position paper analyzes the important role that EAs can play in the design and enhancement of GPAIS. It starts with a background of the benefits of Evolutionary Computation and ML, GPAIS definition and properties, and the importance of the paradigm of *AI-powered AI* for GPAIS. Then, by leveraging this paradigm adapted for EAs, this work shows how *EA-powered AI* can be used to both design or enhance GPAIS. It follows with the matching between the core properties of GPAIS and the EA-based research areas that could be beneficial to realize them. Based on the definition of GPAIS presented in this position paper, various recent milestones of closed- and open-world EA-GPAIS are described. Lastly, it concludes with the challenges of harnessing the benefits of EA-GPAIS and with several strategies that can be implemented with EAs to address advances for the design and enhancement of GPAIS.

The position paper offers valuable insights:

- Provide a comprehensive work about the role of EAs in the emerging field of GPAIS.
- Identify how *EA-powered AI* can be used to design or enhance GPAIS.
- Match the potential of EA-based areas to realize GPAIS properties.

- Indicate strategies to explore promising research areas of EAs that can be useful in GPAIS.

This study has provided several lessons about the importance of EAs in the design and enhancement of GPAIS. Next, we highlight the following ones:

- The emerging field of GPAIS needs optimization processes. The incorporation in GPAIS of ideas and algorithms like EAs, which have contributed to the development of the field of ML, can be useful.
- The landscape of EAs application is larger, as shown in recent EA-GPAIS. The ability of MOEAs to explore several objectives allows the incorporation of diversity measures. This feature facilitates a natural integration into more open-world GPAIS.
- The advancement of GPAIS is the future of AI. These systems hold immense promise for the field. Hence, tools such as EAs, known for their ease of implementation and inherent adaptability and robustness, emerge as an excellent choice for the design and optimization of GPAIS.

7 Conclusions and Future Work

This section concludes the thesis (Subsection 7.1), gathers all the relevant studies we have published (Subsection 7.2), and provides notes on future research lines (Subsection 7.3).

7.1 Conclusions

This thesis presents a comprehensive study of EAs that provides both a comprehensive view of the work already done in the area and innovation in the form of two proposals based on EAs and a position paper. This thesis aims to expand the knowledge about EAs and approach the problem from new perspectives. To this end, the most comprehensive survey on bio-inspired algorithms in the literature has been carried out, extensive experimental studies have also been conducted to prove the potential of our proposals to achieve higher standards than the previously published alternatives, and a position paper that explores the potential of EAs in the field of GPAIS has also been developed.

To achieve the first objective of developing a survey about MHs, an extensive study about MHs with more than 500 reviewed algorithms has been collected. We introduce all the necessary concepts about MHs to fully understand all its fundamentals. We propose a dual taxonomy of both biological inspiration and mathematical behavior, covering the literature of MHs. This proposal presents two novelties: an extended biological taxonomy with more categories and a novel taxonomy based on the underlying mathematical model. We also analyze the influential MHs over the history of the field and the relation of these MHs with the rest of the proposals of the study. Both analyses lead to several lessons learned about the situation of the MHs field, which cover better comparisons, natural inspiration is less relevant than behavior, many proposals with limited influence, and that nature-based terminology can make it difficult to understand novel MHs. We examine the present and future situation of the field of MHs, analyzing its shortcomings, benefits, and potential future applications, revealing promising areas of research like Generative AI. Finally, we provide a brief explanation of recent taxonomies, overviews, and general approaches, which could be useful to other researchers.

The second objective has been addressed with the development of EvoPruneDeepTL. The main contribution of EvoPruneDeepTL is the design of DL models using a binary EA where unnecessary neurons are removed, together with a refinement process of the TL paradigm. We have developed an EA capable of designing robust and improved networks in different domains and also, as a complementary detail, with fewer connections. Moreover, the comparison with other competitive pruning methods able to reduce as much as EvoPruneDeepTL the fully connected layers do not reach the same performance level as our proposal. The experiments to test the generalization ability of EvoPruneDeepTL also achieved good results. EvoPruneDeepTL establishes a new line of research in the design of DL models, using pruning to remove unnecessary neurons to adapt the network to the problem at hand, with the main contribution of improving the performance of such

a network allowing the use of TL.

The third objective was achieved with the development of MOEvoPruneDeepTL. To fulfill this objective, we propose a MOEA for the optimization of three objectives: performance, complexity, and robustness. The generated models are compared in the PF of the solutions, resulting in diverse solutions that achieve high values of performance and robustness and low values of complexity. Also, thanks to the diversity of the models, the ensemble approach can improve the models. The study of the most influential neurons reveals common neurons between solutions, thus discovering certain patterns. These neurons are reflected in the region of the original image, so we can identify the region that is connected to them. Studying the most influential neurons explains the relationship between the neurons and the original image.

The last objective has been tackled with a position paper about the important role of EAs in GPAIS. These algorithms are one of the most important families of bio-inspired algorithms and have achieved great success in ML. Their application to other areas of AI constitutes a promising avenue. *EA-powered AI* has shown how EAs can help to design or enhance GPAIS. The connection of GPAIS properties to EA-based research areas has unveiled the capability of these algorithms for their fulfillment. Recent milestones regarding EA-GPAIS have been presented, showing the advancements in the field. The challenges to obtain the benefits of using EAs in GPAIS scenarios have been revisited. Finally, strategies that can be implemented with EAs to address advances in GPAIS have discovered several research areas that can potentially be very beneficial in the forthcoming years. This objective shows the present and future applications of EAs to innovate and revolutionize the field of GPAIS and the AI.

Conclusiones

Esta tesis presenta un amplio estudio en MHs que proporciona tanto una visión global de los trabajos ya realizados en el área como un enfoque más innovador con dos propuestas basadas en EAs y un trabajo de posicionamiento. El objetivo general de esta tesis es ampliar el conocimiento actual sobre MHs, en particular sobre EAs, y estudiar este problema desde diferentes y nuevas perspectivas. Para ello, se han llevado a cabo la revisión bibliográfica sobre algoritmos bioinspirados más extensa hasta la fecha, estudios experimentales exhaustivos para demostrar el potencial de nuestras propuestas y superar los resultados de las alternativas existentes, y un trabajo de posicionamiento que explora el potencial de los EAs dentro del campo de los GPAIS.

Para realizar el primer objetivo de desarrollar una revisión bibliográfica exhaustiva sobre las MHs, se ha recopilado y analizado un extenso corpus bibliográfico sobre MHs con más de 500 propuestas revisadas. Primero, se introducen los conceptos necesarios sobre las MHs para poder entender sus nociones básicas. Tras ello, se propone una doble taxonomía que abarca la literatura de MHs basada en la inspiración biológica y en la matemática del modelo. Esta propuesta ha presentado dos novedades: una taxonomía biológica con un mayor número de categorías y una nueva taxonomía basada en el modelo

matemático subyacente. Además, se realiza un análisis sobre las MHs más importantes de la historia del campo y su relación con el resto de MHs presentadas en este estudio. Ambos análisis conducen a varias lecciones aprendidas sobre la situación del campo de la MHs, que abarcan mejores comparaciones, que la inspiración natural es menos relevante que el comportamiento, que hay muchas propuestas con influencia limitada, y que la terminología basada en la naturaleza puede dificultar la comprensión de la nueva MHs. Examinamos la situación actual y futura del campo de la MHs, analizando sus limitaciones, beneficios y posibles aplicaciones futuras, desvelando áreas de investigación prometedoras como la *Generative AI* (IA Generativa). Por último, ofrecemos una breve explicación de varias guías de trabajo, taxonomías recientes, trabajos de revisión y enfoques generales, que podrían ser útiles para otros investigadores.

El segundo objetivo se alcanza con el desarrollo de EvoPruneDeepTL. El principal aporte de EvoPruneDeepTL es el diseño de redes neuronales usando un EA binario con el que se eliminan neuronas innecesarias, junto con un refinamiento del proceso de TL. Se ha desarrollado un EA capaz de diseñar redes robustas y mejoras en diferentes problemas y, a modo complementario, con menos conexiones. La comparación con otros modelos de poda competitivos capaces de reducir tanto como EvoPruneDeepTL no obtienen un rendimiento similar que nuestra propuesta. Los experimentos que permiten comprobar la capacidad de generalización de EvoPruneDeepTL también presentan unos buenos resultados. EvoPruneDeepTL establece una nueva línea de investigación en el diseño de modelos DL, utilizando la técnica de la poda para eliminar neuronas innecesarias con el fin de adaptar la red al problema planteado, con la principal aportación de mejorar el rendimiento de dicha red permitiendo el uso del TL.

El tercer objetivo se ha alcanzado con el desarrollo de MOEvoPruneDeepTL. La investigación que se ha llevado a cabo para realizar este objetivo propone un MOEA para la optimización de tres objetivos: rendimiento, complejidad y robustez. Los modelos generados se comparan en el frente de Pareto de las mejores soluciones, obteniéndose soluciones diversas que alcanzan altos valores de rendimiento y robustez y bajos valores de complejidad. Además, gracias a la diversidad de dichos modelos, el enfoque de *ensemble* es capaz de mejorar los modelos. Por último, el estudio de las neuronas más influyentes permite encontrar neuronas comunes entre las soluciones, descubriendo así ciertos patrones. Estas neuronas se reflejan en la región de la imagen original, por lo que es posible identificar la región que está conectada a ellas. Gracias al estudio de las neuronas más influyentes podemos explicar la representación entre dichas neuronas y la imagen original.

El último objetivo se ha abordado con un trabajo de posicionamiento sobre el papel de los EAs para los GPAIS. Estos algoritmos son una de las familias más importantes de algoritmos bioinspirados y han alcanzado un gran éxito en el ML. Su aplicación a otras áreas de la AI constituye una línea de trabajo prometedora. El *EA-powered AI* ha mostrado cómo los EAs pueden ayudar a diseñar o mejorar los GPAIS. La conexión de las propiedades de los GPAIS con las áreas de investigación basadas en EA ha desvelado la capacidad de estos algoritmos para su cumplimiento. Se han presentado hitos recientes en relación con EA-GPAIS, mostrando los avances en el campo. Finalmente, las estrategias

que pueden ser implementadas con los EAs para abordar los avances en los GPAIS han descubierto varias áreas de investigación que potencialmente pueden ser muy beneficiosas en los próximos años. Este objetivo muestra las aplicaciones presentes y futuras de los EAs para innovar y revolucionar el campo de los GPAIS y de la AI.

7.2 Publications

This section lists the journal papers published during the PhD study period, ordered by publishing date. For each publication, we show the DOI, the number of citations, and the Field-Weighted Citation Impact (FWCI) given by Scopus, which shows how well-cited this document is when compared to similar documents. A value greater than 1 means the document is more cited than expected according to the average. This metric takes into account (1) the year of publication, (2) the document type, and (3) the disciplines associated with its source. The list of papers is listed as follows:

- **Journal papers:**

  1. Molina, D., Poyatos, J., Del Ser, J., García, S., Hussain, A., & Herrera, F. (2020). Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. Cognitive Computation, 12, 897-939. DOI: `https://doi.org/10.1007/s12559-020-09730-8`. CITED BY: 108. FWCI: 6.41.
  2. Poyatos, J., Molina, D., Martinez, A. D., Del Ser, J., & Herrera, F. (2023). EvoPruneDeepTL: An evolutionary pruning model for transfer learning based deep neural networks. Neural Networks, 158, 59-82. DOI: `https://doi.org/10.1016/j.neunet.2022.10.011`. CITED BY: 9. FWCI: 4.03.
  3. Poyatos, J., Molina, D., Martínez-Seras, A., Del Ser, J., & Herrera, F. (2023). Multiobjective evolutionary pruning of Deep Neural Networks with Transfer Learning for improving their performance and robustness. Applied Soft Computing, 147, 110757. DOI: `https://doi.org/10.1016/j.asoc.2023.110757`. CITED BY: 3. FWCI: 1.36.

7.3 Future work

The results of this PhD thesis open new lines of research, including the identification of new challenges in the field of EAs. This last section presents future work and lines of research that emerge from the studies and conclusions of this thesis:

Extending the evolutionary search to higher layers of the neural hierarchy. This future work follows naturally from the work presented in EvoPruneDeepTL. Our study focuses on the specialization of the final layers using TL for the convolutional layer weights, but it is also possible to proceed at the level of the convolutional layer

configuration. Once the weights have been determined using TL, it might be useful to use a EAs to evolve the initial weights for each layer towards the optimal set of weights. The TL would serve as a warm start of the algorithm, and local search could even be incorporated to improve these weights in each layer.

Optimization of new objectives. EAs are search algorithms designed to optimize a fitness function. Most proposals focus exclusively on performance and, at most, incorporate another objective in multi-objective cases. However, recent research has shown a growing interest in combining EAs with metrics that measure diversity and robustness, such as OoD detection. This thesis asserts the benefits of combining these techniques. Therefore, this future work will focus on using metrics to measure the desirable properties of the models and will guide the process of finding the best models for the problem at hand. This work would complement other approaches discussed in this section, and their combination could lead to innovative solutions for the field.

Application of distributed co-evolutionary algorithms. In both practical proposals, we have designed and improved the fully-connected layers of the model. However, optimizing the entire model is a major challenge. For this, we could optimize the different parts of the neural network with a distributed approach, using distributed co-evolutionary EAs, so that we could have different populations to optimize the network in a distributed way, thus being able to reduce the optimization time.

Updating MHs libraries for more engagement. The availability of software is important for making comparisons between algorithms efficiently, effectively, and reproducibly, as we have freed the software of our two practical proposals, following the recommendations of open science. Nowadays, a wide variety of tools are available, ranging from data manipulation to statistical tests. However, few studies provide the code associated with the results related to their MH, allowing subsequent comparison with other similar proposals in terms of their biological nature or mathematical behavior. Although there are remarkable MHs libraries that gather a wide range of algorithms [3] [4], a great contribution could be the update of them with the reviewed algorithms of our survey. In this scenario, these libraries could benefit researchers by providing them with more possible algorithms to compare. Our survey will serve as a guide to compare algorithms based on their biological and mathematical characteristics of the algorithm. This would facilitate both the research and development of new algorithms as well as a fair comparison between them, making them more accessible to researchers in the field.

Trustworthiness, robustness, security, and safe systems. The design of any system should aim to achieve certain properties to ensure correct operation. This thesis illustrates how to achieve robustness in the two practical works. Achieving the remaining properties for the different models developed presents a double challenge. These challenges involve, in the first instance, recognizing the scenarios where the model needs to detect possible

[3] `https://github.com/thieu1995/mealpy`

[4] `http://esa.github.io/pygmo/index.html`

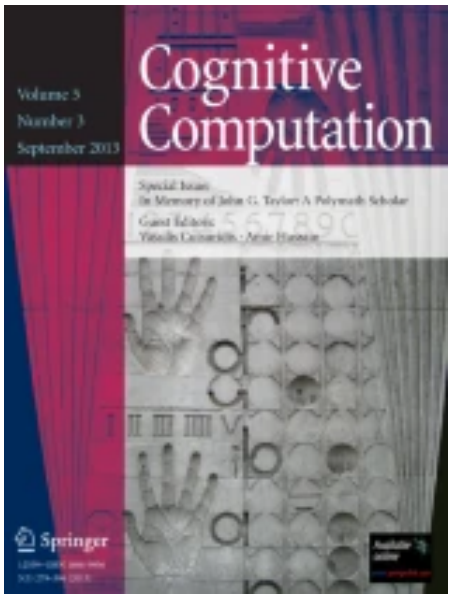
failures and, in the second phase, exploring the feasibility of evolutionary strategies to address these failures.

Development of an open-world GPAIS. Within the DL field, a variety of closed-world GPAIS have been created, but in recent years new proposals based on open-world ideas have appeared [WLCS19, MLW+23]. Some of those ideas, such as knowledge transfer, robustness based on new samples (OoD), and ensemble learning, have been used in this thesis. This new research line would aim to continue the use of EAs taking into account the different ways to generate diversity and to develop an open-world system so that the system would focus not only on knowing how to perform the tasks it knows, but also on adapting to perform new ones.

Chapter II

Publications

1 Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations



- Journal: Cognitive Computation
- JCR Impact Factor: 5.418
- Rank: 33/139
- Quartile: Q1
- Category: Computer Science, Artificial Intelligence
- Status: Published

Ref.: Molina, D., Poyatos, J., Ser, J. D., García, S., Hussain, A., & Herrera, F. (2020). Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. Cognitive Computation, 12, 897-939. DOI: https://doi.org/10.1007/s12559-020-09730-8

Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration versus Algorithmic Behavior, Critical Analysis and Recommendations (from 2020 to 2024)

Daniel Molina[1], **Javier Poyatos**[1], **Javier Del Ser**[2,3], **Salvador García**[1], **Amir Hussain**[4], and **Francisco Herrera**[1]

[1] Department of Computer Science and Artificial Intelligence, DaSCI Andalusian Institute of Data Science and Computation Intelligence, University of Granada, Spain, {dmolina, savalgl, herrera}@decsai.ugr.es, jpoyatosamador@ugr.es
[2] TECNALIA, Basque Research & Technology Alliance (BRTA), Spain, javier.delser@tecnalia.com
[3] Dept. of Communications Engineering, University of the Basque Country (UPV/EHU), Spain
[4] Edinburgh Napier University, United Kingdom, a.hussain@napier.ac.uk

Abstract

In recent years, bio-inspired optimization has garnered significant attention in the literature. This algorithmic family mimics various biological processes observed in nature to effectively tackle complex optimization problems. The proliferation of nature- and bio-inspired algorithms, accompanied by a plethora of applications, tools, and guidelines, underscores the growing interest in this field. However, the exponential rise in the number of bio-inspired algorithms poses a challenge to the future trajectory of this research domain. Along the five versions of this document, the number of approaches grows incessantly, and where having a new biological description takes precedence over real problem-solving. This document, in its fifth revision since the original published version in [1], presents two comprehensive taxonomies. One is based on principles of biological similarity, and the other one is based on operational aspects associated with the iteration of population models that initially have a biological inspiration. Therefore, these taxonomies enable researchers to categorize existing algorithmic developments into well-defined classes, considering two criteria: the source of inspiration and the behavior exhibited by each algorithm. Using these taxonomies, we classify 518 algorithms based on nature-inspired and bio-inspired principles. Each algorithm within these categories is thoroughly examined, allowing for a critical synthesis of design trends and similarities, and identifying the most analogous classical algorithm for each proposal. From our analysis, we conclude that a poor relationship is often found between the natural inspiration of an algorithm and its behavior. Furthermore, similarities in terms of behavior between different algorithms are greater than what is claimed in their public disclosure: specifically, we show that more than one-fourth of the reviewed bio-inspired solvers are versions of classical algorithms. The conclusions from the analysis of the algorithms lead to several learned lessons.

Moreover, in this new update we have decided to take a brief tour of literature towards three broad directions, providing a more extensive approach to the original document:

- First, we offer a critical perspective on the field following our insights in [2], highlighting the *good* (a present and future plenty of exciting applications), the *bad* (novel metaphors not leading to innovative solvers), and the *ugly* (poor methodological practices) in metaheuristic optimization, with an expansion of these perspectives.
- Second, we revisit evolutionary and bio-inspired algorithms from a threefold approach: i) where we stand and what's next in evolutionary algorithms and population-based nature and bio-inspired optimization, based on a structured proposal of challenges that were discussed in 2020, but still exist today [3]; ii) a prescription of methodological guidelines for comparing bio-inspired optimization algorithms [4]; and iii) a tutorial on the design, experimentation, and application of metaheuristic algorithms to real-world optimization problems [5].
- Third, we perform a brief review of recent studies that propose good practices for designing metaheuristic algorithms, alongside a few highlighted taxonomies, overviews, and general approaches that, far from without attempting to be exhaustive with the literature, showcase the rich activity and attention received by this field in recent years.

This updated study ends with an analysis that exposes the double vision of the wide range of proposals that contributed to the field of metaheuristic optimization after five years of analysis: on one hand, we note a lack of analysis of the real optimization challenges and useful proposals instead of new metaheuristics only focused on a basic comparison with very

classical problems versus algorithms. On the other hand, we offer a positive vision of the crucial role that population-based optimization models can take in the design of modern Artificial Intelligence algorithms.

This document is an update as of April 2024, and contains 518 algorithms as opposed to the originally published version which amounted to 323 revised metaheuristics. This arXiv document corresponds with an extension (already mentioned) of the 2 following papers, with references:

- Molina, D., Poyatos, J., Del Ser, J., García, S., Hussain, A., & Herrera, F. (2020). Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. Cognitive Computation, 12, 897-939. DOI: https://doi.org/10.1007/s12559-020-09730-8
- Molina-Cabrera, D., Poyatos, J., Osaba, E., Del Ser, J., Herrera, F. (2022). Nature- and Bio-inspired Optimization: the Good, the Bad, the Ugly and the Hopeful. DYNA, 97(2). 114-117. DOI: https://doi.org/10.6036/10331

Keywords – Nature-inspired algorithms, bio-inspired optimization, taxonomy, critical analysis.

1 Introduction

Traditional optimization techniques are motivated by the complexity of the problem and the mathematical properties of its fitness function and constraints. However, in many real-world optimization problems, no exact solver can be applied to solve them at an affordable computational cost or within a reasonable time. Moreover, in some cases, there is no analytical form for the problem's objective and constraints. Under such circumstances, the use of traditional techniques has been widely proven to be unsuccessful, thereby calling for the consideration of alternative optimization approaches.

In this context, complexity is not unusual in Nature: a plethora of complex systems, processes and behaviors have evinced a surprising performance to efficiently address intricate optimization tasks. The most clear example can be found in the different animal species, which have developed over generations very specialized capabilities by evolutionary mechanisms. Indeed, evolution has allowed animals to adapt to harsh environments, foraging, very difficult tasks of orientation, and to resiliently withstand radical climatic changes, among other threats. Animals, when organized in independent systems, groups or swarms or colonies (systems quite complex on their own) have managed to colonize the Earth completely, and eventually achieve a global equilibrium that has permitted them to endure for thousands of years. This renowned success of biological organisms has inspired all kinds of solvers for optimization problems, which have been so far referred to as *bio-inspired optimization algorithms*. This family of optimization methods simulates biological processes such as natural evolution, where solutions are represented by individuals that reproduce and mutate to generate new, potentially improved candidate solutions for the problem at hand.

Disregarding their source of inspiration, there is clear evidence of the increasing popularity and notoriety gained by nature- and bio-inspired optimization algorithms in the last two decades. This momentum finds its reason in the capability of these algorithms to learn, adapt, and provide good solutions to complex problems that otherwise would have remained unsolved. Many overviews have capitalized on this spectrum of algorithms applied to a wide range of problem casuistry, from combinatorial problems [6] to large-scale optimization [7], evolutionary deep learning [8] and other alike. As a result, almost all business sectors have leveraged this success in recent times.

From a design perspective, nature- and bio-inspired optimization algorithms are usually conceived after observing a natural process or the behavioral patterns of biological organisms, which are then converted into a computational optimization algorithm. New discoveries in Nature and the undoubted increase of worldwide investigation efforts have ignited the interest of the research community in biological processes and their extrapolation to computational problems. As a result, many new bio-inspired meta-heuristics have appeared in the literature, increasing the outbreak of proposals and applications every year. Nowadays, every natural process can be thought to be adaptable and emulated to produce a new meta-heuristic approach, yet with different capabilities of reaching global optimum solutions to optimization problems.

The above statement is quantitatively supported by Figure 1, which depicts the increasing number of papers/book chapters published in the last years with *bio-inspired optimization* and *nature-inspired optimization* in their title, abstract and/or keywords. We have considered both *bio-inspired* and *nature-inspired optimization* because sometimes both terms are confused and indistinctly used, although nature-inspiration includes bio-inspired inspiration and complements it with other sources of inspirations (like physics-based optimization, chemistry-based optimization, ...). A major fraction of the publications

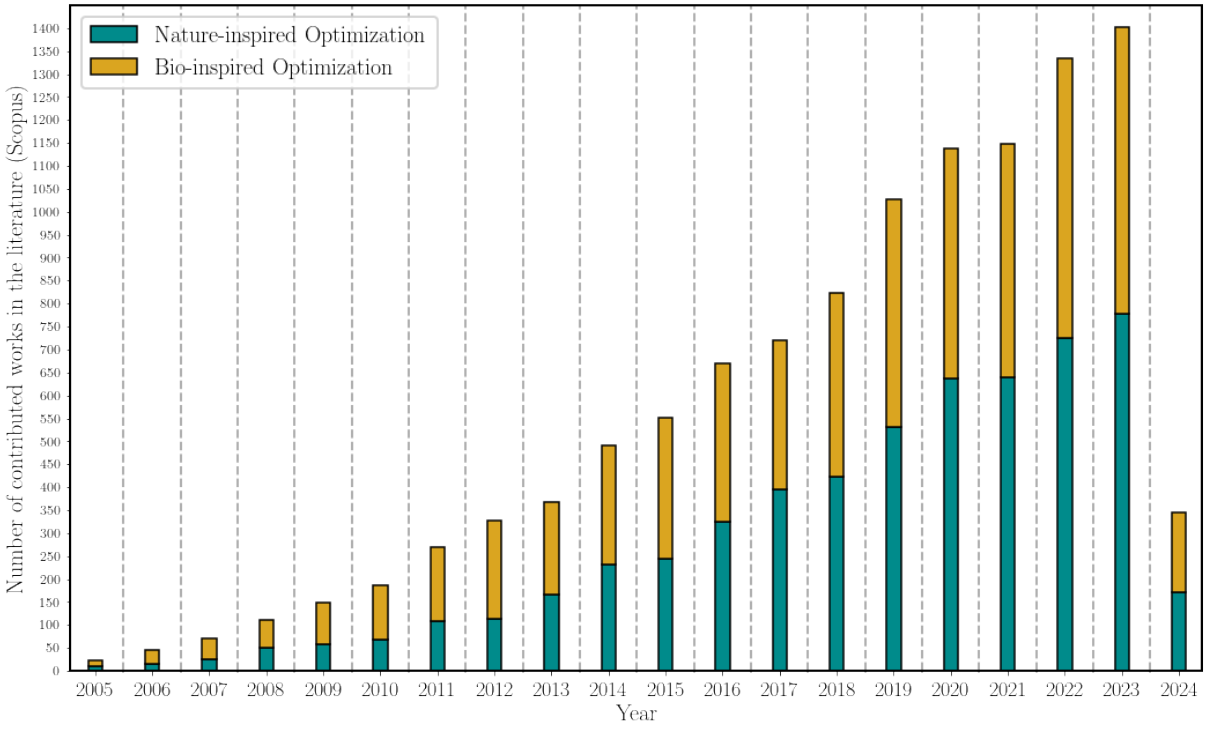


Figure 1: Number of papers with *bio-inspired optimization* and *nature-inspired optimization* in the title, abstract and/or keywords, over the period 2005–April 2024 (Scopus database).

comprising this plot proposed new bio-inspired algorithms at their time. From this rising number of nature and bio-inspired algorithms, one can easily conclude that it would be convenient to organize them into a taxonomy with well-defined criteria where to classify both the existing bio-inspired algorithms and new proposals to appear in the future. Unfortunately, the majority of such publications do not include any type of taxonomy, nor do they perform an exhaustive analysis of the similarity of their proposed algorithms concerning other counterparts. Instead, they only focus on the nature or biological metaphor motivating the design of their meta-heuristic.

This metaphor-driven research trend has been already underscored in several contributions [9, 10], which have unleashed hot debates around specific meta-heuristic schemes that remain unresolved to date [11, 12]. This problem gets exacerbated when important challenges are overseen and if more and more biological inspirations are used as the primary driver for research, as we can observe in 2024 with more than 500 proposals. It is our firm belief that this controversy could be lessened by a comprehensive taxonomy of nature and bio-inspired optimization algorithms that settled the criteria to justify the novelty and true contributions of current and future advances in the field.

In this fifth version of the original study published in [1], we have classified 518 works proposing different types of meta-heuristic algorithms. Building upon this knowledge, we herein propose two different taxonomies for nature- and bio-inspired optimization algorithms:

- The first taxonomy classifies algorithms based on their natural or biological inspiration so that given a specific algorithm, we can find its category quickly and without any ambiguity. The goal of this first taxonomy is to allow easy grouping the upsurge of solvers published in the literature.

- The second taxonomy classifies the reviewed algorithms based exclusively on their behavior, i.e., how they generate new candidate solutions for the function to be optimized. Our aim is to group together algorithms with similar behavior, without

considering its inspirational metaphor.

We believe that this dual criterion can be very useful for researchers. The first one helps classify the different proposals by their origin of inspiration, whereas the second one provides valuable information about their algorithmic similarities and differences. This double classification allows researchers to identify each new proposal in an adequate context. To the best of our knowledge, there has been no previous attempt as ambitious as the one presented in this overview to organize the existing literature on nature- and bio-inspired optimization.

Considering the classifications obtained in our study, we have critically examined the reviewed literature classification in the different taxonomies proposed in this work. The goal is to analyze if there is a relationship between the algorithms classified in the same category in one taxonomy and their classification in the other taxonomy. Next, similarities detected among algorithms will allow discovering the most influential meta-heuristic algorithms, whose behavior has inspired many other nature- and bio-inspired proposals.

These previous research tasks provide several insights to conduct a comprehensive two-fold analysis of the field:

- The first analysis focuses on taxonomies. Specifically, we provide several recommendations to improve research practices in this area. The growing number of nature-inspired proposals could be seen as a symptom of the active status of this field; however, its sharp evolution suggests that research efforts should be also invested towards new behavioral differences and verifiable performance evidence in practical problems.

- The second analysis delves into a critical perspective on bio-inspired optimization. It discusses the strengths, weaknesses, and challenges that have been identified in the field in recent years, while it also highlights the potential held for future developments in bio-inspired optimization.

Both taxonomies and the analysis provide a full overview of the situation of the bio-inspired optimization field. However, Figure 1 reflects the interest of research in this field, as the number of papers is in continuous growth of interest. We believe that it is essential to highlight and reflect on what is expected from this field in the coming years, in terms of where it is currently being used and how researchers are proposing methodologies to properly design and apply bio-inspired algorithms not in real-world applications, but also in other emerging areas of Artificial Intelligence (AI). As a consequence, an analysis of the field in terms of *Bio-inspired Optimization*, *Evolutionary Computation*, *Guidelines*, *Comparison Methodology* and *Benchmarking* are found in this report.

As we have mentioned in the abstract, in this final version of the report we have decided to take a brief tour of literature from three broad perspectives with a more extensive approach to the document:

In Section 7, we pay attention from a triple critical position as it was pointed out in [2], highlighting the *good* (a present and future plenty of exciting applications), the *bad* (novel metaphors not leading to innovative solvers, going deeper into the group of works that criticize the lack of novelty of the new proposals [13, 9, 11, 10, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]), and the *ugly* (poor methodological practices) as it was pointed out in [2], with an expansion of these analyses. As we have mentioned, we must emphasize that in these new algorithms, there exists a lack of justification together with the lack of comparison with the state of the art and of real interest in achieving reasonable levels of quality from the point of view of the optimization of well-known problems in recent competitions. Good methodological practices must be followed in forthcoming studies when designing, describing, and comparing new algorithms.

The analysis of the issues undergone by the field enables us to provide potential solutions and an analysis toward best practices. Hence, in Section 8, we introduce three previous works, outlined as follows:

- Bio-inspired computation: Where we stand and what's next [3].

- A prescription of methodological guidelines for comparing bio-inspired optimization algorithms [4].

- A tutorial on the design, experimentation, and application of metaheuristic algorithms to real-world optimization problems [5].

Lastly, Section 9 presents an analysis of metaheuristics based on studies, guidelines, and other works of a more theoretical nature that help to solve the problems of the field. We perform a brief review of recent studies that address good practices for designing metaheuristics and discussions from this perspective, and a short review of references – without attempting to be exhaustive – that address taxonomies, overviews, and general approaches in bio-inspired optimization. Therefore, this section considers such studies from a double vision:

- Good practices for designing metaheuristics: It gathers several works that are guidelines for good practices related to research orientation to measure novelty [26], to measure similarity in metaheuristics [27], Metaheuristics "In the Large" (to support the development, analysis, and comparison of new approaches) [28], to design manual or automatic new metaheuristics [29], to guide the learning strategy in design and improvement of metaheuristics [30], to use statistical test in metaheuristics [31], and to detect the novelties in metaphor-based algorithms [32].
- Latest metaheuristics based studies which include, non-exhaustively, a dozen of recent studies about taxonomies [33, 34, 35], overviews [15, 36, 37, 38, 39, 40] and general approaches [41].

This work has been updated almost every year with several improvements, as shown in the trace of changes shown in Table 1. The latest version includes both novel bio-inspired proposals up to April 2024 and several analyses of the field, ranging from the situation of the field to the vision towards the future of this field.

Table 1: Updates of the manuscript via arXiv.

Update	Date	Contribution
Version # 1	Feb. 2020	Initial version of the manuscript with 323 reviewed algorithms.
Version # 2	Feb. 2020	Changes in title and figures for better quality.
Version # 3	Apr. 2021	Update with +31 new algorithms (up to 361), figures and tables changed.
Version # 4	May 2022	Update with +51 new algorithms (up to 412), figures and tables changed.
Version # 5	Apr. 2024	Update with +88 new algorithms (up to 518); figures and tables changed, and three analyses included as Sections 7, 8 and 9.

As we have mentioned in the abstract, this fifth and last version of this series of documents ends with an analysis that addresses the double vision of a wide range of proposals, which after five years of analysis must be indicated that they border on a lack of analysis of the real problems and useful proposals, and on the other hand, a positive vision of the role that population-based optimization models can contribute in the design of AI systems, in a new scenario of continuous emergence of AI.

The rest of this paper is organized as follows. In Section 2, we examine previous surveys, taxonomies, and reviews of nature- and bio-inspired algorithms reported so far in the literature. Section 3 delves into the taxonomy based on the inspiration of the algorithms. In Section 4, we present and populate the taxonomy based on the behavior of the algorithm. In Section 5, we analyze similarities and differences found between both taxonomies, ultimately identifying the most influential algorithms in our reviewed papers. In Section 6, we report several lessons learned and recommendations as the result of the previous analysis. In addition, as novel contributions of this version over its preceding ones, Section 7 provides an extended critical analysis of the state of the art in the field, highlighting the aforementioned *good*, the *bad*, and the *ugly* in the metaheuristic landscape [2]. In Section 8, we discuss future directions in bio-inspired optimization algorithms, and prescribe potential solutions and analysis toward ensuring good practices and correct experimental procedures with these algorithms. Section 9 shows studies and guidelines for good practices, together with recent studies including taxonomies, overviews, and general approaches related to metaheuristics. Finally, in Section 10, we summarize our current main conclusions and reflections on the field, with builds upon a five-year reflection and literature study.

2 Related Literature Studies (before 2020 according to the first version of this report, Feb. 2020)

The diversity of bio-inspired algorithms and their flexibility to tackle optimization problems for many research fields have inspired several surveys and overviews to date. Most of them have focused on different types of problems [42, 43], including continuous [44], combinatorial [6], or multi-objective optimization problems [45]. For specific real-world problems, the prolific literature about nature- and bio-inspired algorithms has sparked specific state-of-the-art studies revolving around their application to different fields, such as Telecommunications [46], Robotics [47], Data Mining [48], Structural Engineering [45] or Transportation [49]. Even specific real-world problems have been dedicated exclusive overviews due to the vast research

reported around the topic, like power systems [50], the design of computer networks [51], automatic clustering [52], face recognition [53], molecular docking [54], or intrusion detection [55], to mention a few.

Seen from the algorithmic perspective, many particular bio-inspired solvers have been modified over the years to yield different versions analyzed in surveys devoted to that type of algorithms, from classical approaches such as PSO [56] and DE [57, 58, 59] to more modern ones, e.g., ABC [60, 61], Bacterial Foraging Optimization Algorithm (BFOA, [62]) or the Bat Algorithm [63]. From a more general albeit still algorithmic point of view, [9] explains how the metaphor and the biological idea are used to create a bio-inspired meta-heuristic optimization algorithm. In this study, the reader is also provided with some examples and characteristics of this design process. Books like [64] or [65] show many nature-inspired algorithms. However, they are more focused on describing the different algorithms available in the literature than on classifying and analyzing them in depth.

Several comparison studies among bio-inspired algorithms with very different behaviors can be found in the current literature, which mostly aims at deciding which approach to use for solving a problem. In [66], the popular PSO and DE versions are compared. This research line is followed by [67], which compared the performance of different bio-inspired algorithms, again with prescribing which one to use as its primary goal. More recently, [68] examined the features of several recent bio-inspired algorithms, suggesting, on a concluding note, to which type of problem each of the examined algorithms should be applied. More specific is the work in [69], which compares several different algorithms considering its parallel implementation on GPU devices. More recently, the focus has shifted towards comparing *groups* of algorithms instead of making a comparison between individual solvers: this is the case of [70], which compares Swarm Intelligence and Evolutionary Computation methods in order to assess their properties and behavior (e.g., their convergence speed). Once again, the main purpose of this recent literature strand is to compare bio-inspired algorithms, not to classify them nor to find similarities and design patterns among them. In [71], foraging algorithms (such as the aforementioned BFOA) are compared against other evolutionary algorithms. In that paper, algorithms are classified into two large groups: algorithms *with* and *without* cooperation. In [72, 73], PSO was proven to outperform other bio-inspired approaches (namely, DE, GA and ABC) when used for efficiently training and configuring Echo State Networks.

It has not been until relatively recent times that the community has embraced the need for arranging the myriad of existing bio-inspired algorithms and classifying them under principled, coherent criteria. In 2013, [74] presented a classification of meta-heuristic algorithms as per their biological inspiration that discerned categories with similar approaches in this regard: *Swarm Intelligence*, *Physics and Chemistry Based*, *Bio-inspired algorithms (not SI-based)*, and an *Other algorithms* category. However, the classification given in this paper is not actually hierarchical, so it can not be regarded as a true taxonomy. Another classification was proposed in [75, 76], composed by *Evolution Based Methods*, *Physics Based Methods*, *Swarm Based Methods*, and *Human-Based Methods*. With respect to the preceding classification, a new *Human-Based* category is proposed, which collectively refers to algorithms inspired by human behavior. The classification criteria underneath these categories are used to build up a catalog of more than 50 algorithmic proposals, obtaining similar groups in size. In this case, there is no *miscellaneous* category as in the previous classification. Similarly to [74], categories are disjoint groups without subcategories.

Recently, [77] offers a review of meta-heuristics from the 1970s until 2015, i.e., from the development of neural networks to novel algorithms like Cuckoo Search. Specifically, a broad view of new proposals is given, but without proposing any category. The most recent survey to date is that in [78], in which nature-inspired algorithms are classified not in terms of their source of inspiration, but rather by their behavior. Consequently, algorithms are classified as per three different principles. The first one is *learning behavior*, namely, how solutions are learned from others preceding them. The learning behavior can be individual, local (i.e., only inside a neighborhood), global (between all individuals), and none (no learning). The second principle is *interaction-collective behavior*, denoting whether individuals cooperate or compete between them. The third principle is referred to as *diversification-population control*, by which algorithms are classified based on whether the population has a converging tendency, a diffuse tendency, or no specific tendency. Then, 29 bio-inspired algorithms are classified by each of these principles, and approaches grouped by each principle are experimentally compared.

The prior related work reviewed above indicates that the community widely acknowledges (with more emphasis in recent times) the need for properly organizing the plethora of bio- and nature-inspired algorithms in a coherent taxonomy. However, the majority of them are only focused on the natural inspiration of the algorithms for creating the taxonomy, not giving any attention to their behavior. This aspect is considered in [78], but does not propose a real taxonomy, but rather different independent design principles. On the contrary, as will be next described, our proposed taxonomies consider 1) the source of inspiration; and 2) the procedure by which new solutions are produced over the search process of every algorithm (*behavior*). Furthermore, we note that efforts invested in this regard to date are not up to the level of ambition and thoroughness pursued

in our study. In addition, no study published so far has been specifically devoted to unveiling structural similarities between classical and modern meta-heuristics. There lies the novelty and core contribution of our study, along with the aforementioned novel behavior-based taxonomy.

3 Taxonomy by Source of Inspiration

In this section, we propose a novel taxonomy based on the inspirational source in which nature- and bio-inspired algorithms are claimed to find their design rationale in the literature. This allows classifying the great amount and variety of contributions reported in related fora.

The proposed taxonomy presented in what follows was developed reviewing 518 papers over different years, starting from the most classical proposals in the late 80's (such as Simulated Annealing [79] or PSO [80]) to more novel techniques appearing in the literature until 2024 [81]. Thus, to our knowledge, this exercise can be considered the most exhaustive review in the area to date.

Taking into account all the reviewed papers, we group the proposals therein in a hierarchy of categories. In the hierarchy, not all proposals of a category must fit in one of its subcategories. In our classification, categories lying at the same level are disjoint sets, which means that each proposed algorithm can be only a member of one of these categories. To this end, for each algorithm, we select the category considered to be most suitable considering the nuances of the algorithm that allow us to differentiate it from its remaining counterparts.

Methodologically, a classification of all nature- and bio-inspired algorithms that can be found in the literature can become complicated, considering the different sources of inspiration as biological, physical, human-being, ... In some papers, authors suggest a possible categorization of more well-established groups, but not in all of them. Also, its classification could not be more appropriate and become eventually obsolete, since the number of proposals – and thereby, the diversity of sources of inspiration motivating them – increases over time. Algorithms within each proposed category were selected by their relative importance in the field, considering the number of citations, the number of algorithmic variants that were inspired by that algorithm, and other similar factors.

When establishing a hierarchical classification, it is important to achieve a good trade-off between information and simplicity by the following criteria:

- When to establish a new division of a category into subcategories: a coarse split criterion for the taxonomy can imply categories of little utility for the subsequent analysis, since in that case, the same category would group very different algorithms. On the other hand, a fine-grained taxonomy can produce very complex hierarchies and, therefore, with little usefulness. To keep the taxonomy simple yet informative for our analytical purposes, we decided that a category should have at least four algorithms in order to be kept in the taxonomy. Thus, a category is only decomposed into subcategories if each of them has coherence and a minimum representativeness (as per the number of algorithms it contains).

- Which number of subcategories into which to divide a category: the criterion followed in this regard must produce meaningful subcategories. In order to ensure a reduced number of subcategories, we consider that not all algorithms inside one category must be a member of one of its subcategories. In that way, we avoid introducing mess categories that lack cohesion.

Figure 2 depicts the classification we have reached, indicating, for the 518 reviewed algorithms, the number and ratio of proposals classified in each category and subcategory. It can be observed that the largest group of all is *Swarm Intelligence* category (more than a half of the proposed, 53%), inspired in the Swarm Intelligence concept [64], followed by the *Physics and Chemistry* category, inspired by different physical behaviors or chemical reactions (almost 15% of proposals). Other relevant categories are *Social Human Behavior Algorithms* (11%), inspired by human aspects, and *Breeding-based Evolution* (near 7%), inspired by the Theory of Evolution over a population of individuals, that includes very renowned algorithms such as Genetic Algorithms. A new category emerges from our study – *Plants Based* – which has not been included in other taxonomies. Nearly 10% of the proposals are so different between them that they cannot be grouped in new categories. The percentage of classification of each category is visually displayed in Figure 3.

For the sake of clarity regarding the classification criteria, in the next subsections, we provide a brief description of the different categories in this first taxonomy, including their main characteristics, an example, and a table listing the algorithms belonging to each category.

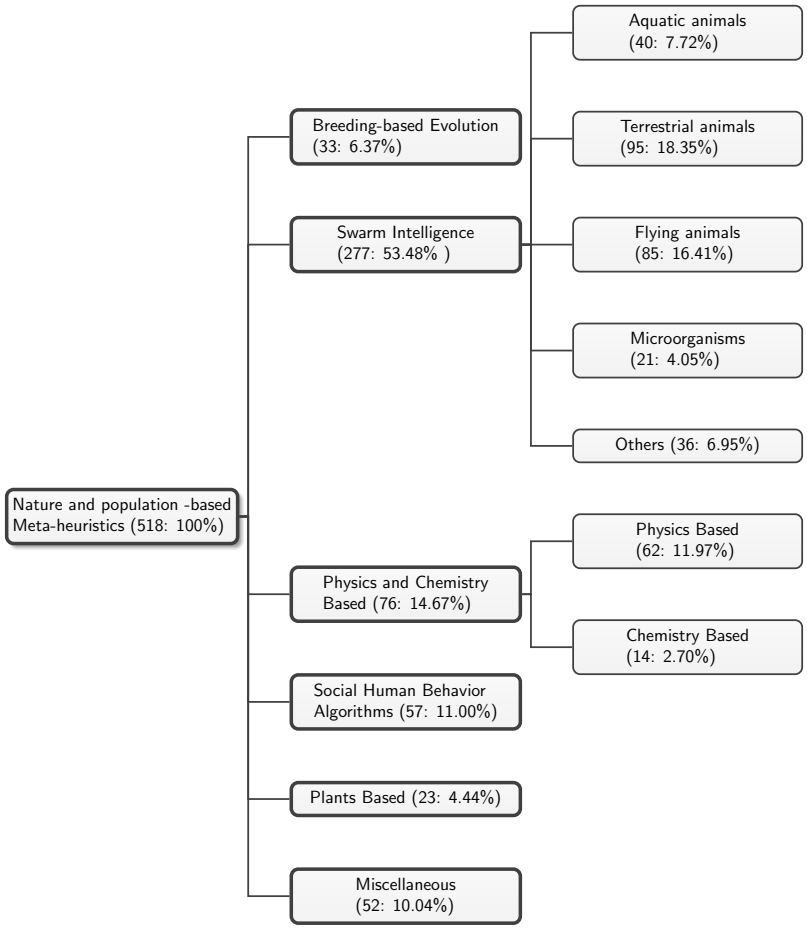


Figure 2: Classification of the reviewed papers using the *inspiration source* based taxonomy.

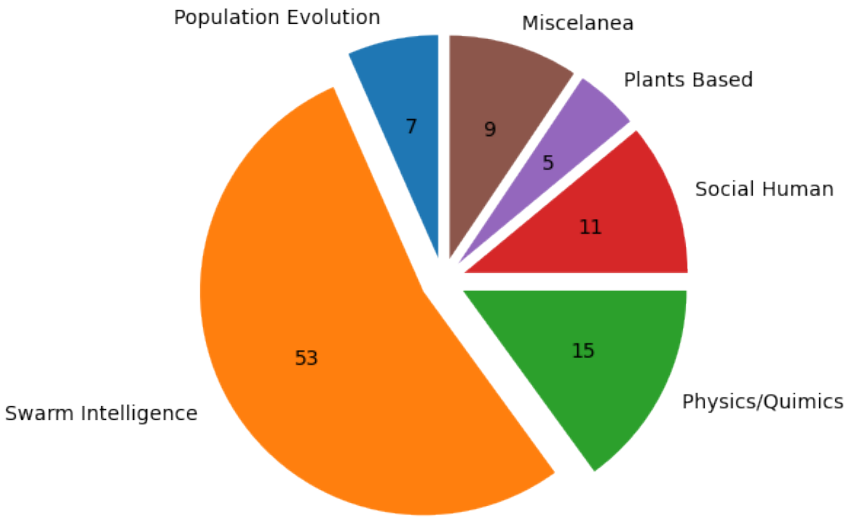


Figure 3: Ratio of reviewed algorithms by its category (first taxonomy).

3.1 Breeding-based Evolutionary Algorithms

This category comprises population-based algorithms inspired by the principles of Natural Evolution. Each individual in the population represents a solution of the problem and has an associated fitness value (namely, the value of the problem objective function for that solution). In these algorithms, a process of reproduction (also referred to breeding or crossover) and survival iterated over successive generations makes the population of solutions potentially evolve towards regions of higher optimality (as told by the best solution in the population). Thus, this category is characterized by the fact of being inspired by the concept of breeding-based evolution, without considering other operators performed on individuals than reproduction (e.g., mutation).

More in detail, in these algorithms we have a population with individuals that have the ability to breed and produce new offspring. Therefore, from the parents, we get children, which introduces some variety with respect to their parents. These characteristics allow them to adapt to the environment which, translated to the optimization realm, permits them to search more efficiently over the solution space of the problem at hand. By virtue of this mechanism, we have a population that evolves through generations and, when combined with a selection (survival) and – possibly – other operators, results are improved. Nevertheless, the breeding characteristic is what makes algorithms within this category unique with respect to those in other categories.

Table 2 compiles all reviewed algorithms that fall within this category. As could have been a priori expected, well-known classical Evolutionary Computation algorithms can be observed in this list, such as Genetic Algorithm (GA), Evolution Strategies (ES), and Differential Evolution (DE). However, other algorithms based on the reproduction of different biological organisms are also notable, such as queen bees and weeds.

3.2 Swarm Intelligence based Algorithms

Swarm Intelligence (SI) is already a consolidated term in the community, which was first introduced by Gerardo Beni and Jing Wang in 1989 [47]. It can be defined as the collective behavior of decentralized, self-organized systems, in either natural or artificial environments. The expression was proposed in the context of robotic systems, but has generalized over the years to denote the emergence of collective intelligence from a group of simple agents, governed by simple behavioral rules. Thus,

Table 2: Nature- and bio-inspired meta-heuristics within the *Breeding-based Evolution* category.

Breeding-based Evolution

Algorithm Name	Acronym	Year	Reference
Artificial Ecosystem Algorithm	AEA	2014	[82]
Artificial Ecosystem Optimizer	AEO	2020	[83]
Artificial Infections Disease Optimization	AIDO	2016	[84]
Asexual Reproduction Optimization	ARO	2010	[85]
Biogeography Based Optimization	BBO	2008	[86]
Bird Mating Optimization	BMO	2014	[87]
Bean Optimization Algorithm	BOA	2011	[88]
Coronavirus Mask Protection Algorithm	CMPA	2023	[89]
Coronavirus Disease Optimization Algorithm	COVIDOA	2022	[90]
Coral Reefs Optimization	CRO	2014	[91]
Dendritic Cells Algorithm	DCA	2005	[92]
Differential Evolution	DE	1997	[93]
Ecogeography-Based Optimization	EBO	2014	[94]
Eco-Inspired Evolutionary Algorithm	EEA	2011	[95]
Earthworm Optimization Algorithm	EOA	2018	[96]
Evolution Strategies	ES	2002	[97]
Genetic Algorithms	GA	1989	[98]
Gene Expression	GE	2001	[99]
Hybrid Rice Optimization	HRO	2016	[100]
Immune-Inspired Computational Intelligence	ICI	2008	[101]
Improved Genetic Immune Algorithm	IGIA	2017	[102]
Weed Colonization Optimization	IWO	2006	[103]
Marriage In Honey Bees Optimization	MHBO	2001	[104]
Mushroom Reproduction Optimization	MRO	2018	[105]
Queen-Bee Evolution	QBE	2003	[106]
SuperBug Algorithm	SuA	2012	[107]
Stem Cells Algorithm	SCA	2011	[108]
Sheep Flock Heredity Model	SFHM	2001	[109]
Swine Influenza Models Based Optimization	SIMBO	2013	[110]
Self-Organizing Migrating Algorithm	SOMA	2004	[111]
T-Cell Immune Algorithm	TCIA	2023	[112]
Variable Mesh Optimization	VMO	2012	[113]
Virulence Optimization Algorithm	VOA	2016	[114]

bio-inspired meta-heuristics based on Swarm Intelligence are those inspired by the collective behavior of animal societies, such as insect colonies or bird flocks, wherein the collective intelligence emerging from the swarm permits to efficiently undertake optimization problems. The seminal bio-inspired algorithm relying on SI concepts was PSO [80], followed shortly thereafter by ACO [115]. These early findings around SI concepts applied to optimization spurred many SI-based algorithms in subsequent years, such as ABC [116] and more recently, Firefly Algorithm (FA, [117]) and Grasshopper Optimization Algorithm (GOA, [118]).

Reviewed algorithms that fall under the Swarm Intelligence umbrella are shown in Tables 3, 4, 5, 6, 7 and 8. This is the most populated category of all our study, characterized by a first category that relates to the type of animal that has inspired each algorithm: as such, we find i) *flying animals*, namely, algorithms inspired in the flying movement of birds and flying animals like insects; ii) *terrestrial animals*, inspired by the foraging and hunter mechanisms of land animals; iii) *aquatic animals*, whose inspiration emerges from the movement of fish schools or other aquatic animals like dolphins; and iv) *microorganisms*, inspired by virus, bacteria, algae and others alike.

Inside each subcategory, we have also distinguished whether they are inspired by the foraging action of the inspired living creature – *Foraging-inspired* is in fact another popular term related to this type of inspiration [119] – or, instead, by its movement patterns in general. When the behavior of the algorithm is inspired by both the movement and the foraging behavior of the animal, it is cataloged as foraging inside our first taxonomy. We will next examine in depth each of these subcategories.

3.2.1 Subcategories of SI based algorithms by the environment

As mentioned previously, the global set of Swarm Intelligence algorithms can be divided as a function of the type of animals. Between the possible categories stemming from this criteria, we have grouped them according to the environmental medium inhibited by the inspiring animal (aquatic, terrestrial or aerial). This criterion is not only very intuitive since it inherits a criterion already applied in animal taxonomies, but it also relies on the fact that these environments condition the movement and hunting mode of the different species. As such, the following subcategories have been established:

- **Flying animals:** This category comprises meta-heuristics based on the concept of Swarm Intelligence in which the trajectory of agents is inspired by flying movements, as those observed in birds, bats, or other flying insects. The most well-known algorithms in this subcategory are PSO [80] and ABC [116].

- **Terrestrial animals:** Meta-heuristics in this category are inspired by foraging or movements in terrestrial animals. The most renowned approach within this category is the classical ACO meta-heuristic [115], which replicates the stigmergic mechanism used by ants to locate food sources and inform of the existence of their counterparts in the colony. This category also includes other popular algorithms like Grey Wolf Optimization (GWO, [240]), inspired in the wild wolf hunting strategy, Lion Optimization Algorithm (LOA, [267]), which imitates hunting methods used by these animals, or the more recent Grasshopper Optimization Algorithm (GOA, [118]), which finds its motivation in the jumping motion of grasshoppers.

- **Aquatic animals:** This type of meta-heuristic algorithm focuses on aquatic animals. The aquatic ecosystem in which they live has inspired different exploration mechanisms. It contains popular algorithms such as Krill Herd (KH, [259]), Whale Optimization Algorithm (WOA, [380]), and algorithms based on the echolocation used by dolphins to detect fish like Dolphin Partner Optimization (DPO, [201]) and Dolphin Echolocation [195].

- **Microorganisms:** Meta-heuristics based on microorganisms are related with the food search performed by bacteria. A bacteria colony performs a movement to search for food. Once they have found and taken it, they split to search again in the environment. Other types of meta-heuristics that can be part of this category are the ones related with virus, which usually replicate the infection process of the cell by virus. The most known algorithm of this category is Bacterial Foraging Optimization Algorithm (BFOA, [148]).

3.2.2 Subcategories of SI based algorithms by the inspirational behavior

Another criterion to group SI based algorithms is the specific behavior of the animal that captured the attention of researchers and inspired the algorithm. This second criterion is also reflected in Tables 3-6, classifying each algorithm as belonging to one of the following behavioral patterns:

Table 3: Nature- and bio-inspired meta-heuristics within the *Swarm Intelligence* category (I).

Swarm Intelligence (I)

Algorithm Name	Acronym	Subcategory	Type	Year	Reference
Artificial Algae Algorithm	AAA	Micro	Movement	2015	[120]
Artificial Beehive Algorithm	ABA	Flying	Foraging	2009	[121]
Artificial Bee Colony	ABC	Flying	Foraging	2007	[116]
Animal Behavior Hunting	ABH	Other	Foraging	2014	[122]
African Buffalo Optimization	ABO	Terrestrial	Foraging	2016	[123]
Andean Condor Algorithm	ACA	Flying	Foraging	2019	[124]
Ant Colony Optimization	ACO	Terrestrial	Foraging	1996	[115]
Artificial Feeding Birds	AFB	Flying	Movement	2018	[125]
Artificial Hummingbird Algorithm	AHA	Flying	Foraging	2022	[126]
Archerfish Hunting Optimizer	AHO	Aquatic	Foraging	2022	[127]
Animal Migration Optimization	AMO	Other	Movement	2014	[128]
Aphid Metaheuristic Optimization	AMO.1	Micro	Movement	2022	[129]
Ant Lion Optimizer	ALO	Terrestrial	Foraging	2015	[130]
Aquila Optimizer	AO	Flying	Foraging	2021	[131]
Anglerfish Algorithm	AOA	Aquatic	Movement	2019	[132]
Arithmetic Optimization Algorithm	AOA.2	Other	Movement	2021	[133]
Artificial Rabbits Optimization	ARO.1	Terrestrial	Foraging	2022	[134]
Artificial Searching Swarm Algorithm	ASSA	Other	Movement	2009	[135]
Artificial Tribe Algorithm	ATA	Other	Movement	2009	[136]
African Wild Dog Algorithm	AWDA	Terrestrial	Foraging	2013	[137]
American Zebra Optimization Algorithm	AZOA	Terrestrial	Movement	2023	[138]
Bald Eagle Search	BES	Flying	Foraging	2019	[139]
Bees Algorithm	BA	Flying	Foraging	2006	[140]
Bumblebees	BB	Flying	Foraging	2009	[141]
Bison Behavior Algorithm	BBA	Terrestrial	Movement	2019	[142]
Bee Colony-Inspired Algorithm	BCIA	Flying	Foraging	2009	[143]
Bee Colony Optimization	BCO	Flying	Foraging	2005	[144]
Bacterial Colony Optimization	BCO.1	Micro	Foraging	2012	[145]
Bacterial Chemotaxis Optimization	BCO.2	Micro	Foraging	2002	[146]
Border Collie Optimization	BCO.3	Terrestrial	Movement	2020	[147]
Biomimicry Of Social Foraging Bacteria for Distributed Optimization	BFOA	Micro	Foraging	2002	[148]
Bacterial Foraging Optimization	BFOA.1	Micro	Foraging	2009	[62]
Bacterial-GA Foraging	BGAF	Micro	Foraging	2007	[149]
BeeHive Algorithm	BHA	Flying	Foraging	2004	[150]
Bees Life Algorithm	BLA	Flying	Foraging	2018	[151]
Bat Intelligence	BI	Flying	Foraging	2012	[152]
Bat Inspired Algorithm	BIA	Flying	Foraging	2010	[153]
Biology Migration Algorithm	BMA	Other	Movement	2019	[154]
Barnacles Mating Optimizer	BMO.1	Micro	Movement	2019	[155]
Blind, Naked Mole-Rats Algorithm	BNMR	Terrestrial	Foraging	2013	[156]
Butterfly Optimizer	BO	Flying	Movement	2015	[157]
Bonobo Optimizer	BO.1	Terrestrial	Movement	2019	[158]
Bull Optimization Algorithm	BOA.1	Terrestrial	Movement	2015	[159]
Bee System	BS	Flying	Foraging	1997	[160]
Bee System	BS.1	Flying	Foraging	2002	[161]
Bird Swarm Algorithm	BSA	Flying	Movement	2016	[162]
Bee Swarm Optimization	BSO	Flying	Foraging	2010	[163]

Table 4: Nature- and bio-inspired meta-heuristics within the *Swarm Intelligence* category (II).

Swarm Intelligence (II)

Algorithm Name	Acronym	Subcategory	Type	Year	Reference
Bioluminiscent Swarm Optimization Algorithm	BSO.1	Flying	Foraging	2011	[164]
Biological Survival Optimizer	BSO.4	Other	Movement	2023	[165]
Bees Swarm Optimization Algorithm	BSOA	Flying	Foraging	2005	[166]
Buzzard Optimization Algorithm	BUZOA	Flying	Foraging	2019	[167]
Black Widow Optimization Algorithm	BWO	Terrestrial	Movement	2020	[168]
Beluga Whale Optimization	BWO.1	Aquatic	Foraging	2022	[169]
Binary Whale Optimization Algorithm	BWOA	Aquatic	Foraging	2019	[170]
Collective Animal Behavior	CAB	Other	Foraging	2012	[171]
Cheetah Based Algorithm	CBA	Terrestrial	Movement	2018	[172]
Catfish Optimization Algorithm	CAO	Aquatic	Movement	2011	[173]
Cricket Behavior-Based Algorithm	CBBE	Terrestrial	Movement	2016	[174]
Cultural Coyote Optimization Algorithm	CCOA	Terrestrial	Movement	2019	[175]
Chaotic Crow Search Algorithm	CCSA	Flying	Foraging	2018	[176]
Chaotic Dragonfly Algorithm	CDA	Flying	Foraging	2018	[177]
Cuttlefish Algorithm	CFA	Aquatic	Movement	2013	[178]
Consultant Guide Search	CGS	Other	Movement	2010	[179]
Camel Herd Algorithm	CHA	Terrestrial	Foraging	2017	[180]
Chimp Optimization Algorithm	ChOA	Terrestrial	Foraging	2020	[181]
Cuckoo Optimization Algorithm	COA	Flying	Foraging	2011	[182]
Camel Travelling Behavior	COA.1	Terrestrial	Movement	2016	[183]
Coyote Optimization Algorithm	COA.2	Terrestrial	Movement	2018	[184]
COOT Optimization Algorithm	COA.5	Flying	Movement	2021	[185]
Coati Optimization Algorithm	COA.6	Terrestrial	Foraging	2023	[186]
Crested Porcupine Optimizer	CPO	Terrestrial	Movement	2024	[187]
Cuckoo Search	CS	Flying	Foraging	2009	[188]
Crow Search Algorithm	CSA	Flying	Movement	2016	[189]
Chameleon Swarm Algorithm	CSA.2	Terrestrial	Foraging	2021	[81]
Circle Search Algorithm	CSA.3	Other	Movement	2022	[190]
Cat Swarm Optimization	CSO	Terrestrial	Movement	2006	[191]
Chicken Swarm Optimization	CSO.1	Terrestrial	Movement	2014	[192]
Dragonfly Algorithm	DA	Flying	Foraging	2016	[193]
Dragonfly Swarm Algorithm	DA.1	Flying	Foraging	2020	[194]
Dolphin Echolocation	DE.1	Aquatic	Foraging	2013	[195]
Dynamic Hunting Leadership	DHL	Other	Foraging	2023	[196]
Deer Hunting Optimization Algorithm	DHOA	Terrestrial	Foraging	2019	[197]
Dwarf Mongoose Optimization	DMO	Terrestrial	Foraging	2022	[198]
Dandelion Optimizer	DO	Other	Movement	2022	[199]
Dingo Optimizer	DOX	Terrestrial	Foraging	2021	[200]
Dolphin Partner Optimization	DPO	Aquatic	Movement	2009	[201]
Donkey Theorem Optimization	DTO	Terrestrial	Foraging	2019	[202]
Enriched Coati Osprey Algorithm	ECOA	Other	Foraging	2024	[203]
Electric Eel Foraging Optimization	EEFO	Aquatic	Foraging	2024	[204]
Electric Fish Optimization	EFO.1	Aquatic	Foraging	2020	[205]
Elephant Herding Optimization	EHO	Terrestrial	Movement	2016	[206]
Elk Herd Optimizer	EHO.1	Terrestrial	Movement	2024	[207]
Ebola Optimization Search Algorithm	EOSA	Micro	Movement	2022	[208]
Emperor Penguins Colony	EPC	Terrestrial	Movement	2019	[209]

Table 5: Nature- and bio-inspired meta-heuristics within the *Swarm Intelligence* category (III).

Swarm Intelligence (III)

Algorithm Name	Acronym	Subcategory	Type	Year	Reference
Emperor Penguin Optimizer	EPO	Terrestrial	Movement	2018	[210]
Eagle Strategy	ES.1	Flying	Foraging	2010	[211]
Elephant Search Algorithm	ESA	Terrestrial	Foraging	2015	[212]
Elephant Swarm Water Search Algorithm	ESWSA	Terrestrial	Movement	2018	[213]
Egyptian Vulture Optimization Algorithm	EV	Flying	Foraging	2013	[214]
Firefly Algorithm	FA	Flying	Foraging	2009	[117]
Flocking Base Algorithms	FBA	Flying	Movement	2006	[215]
Fast Bacterial Swarming Algorithm	FBSA	Micro	Foraging	2008	[216]
Frog Call Inspired Algorithm	FCA	Terrestrial	Movement	2009	[217]
Fire Hawk Optimizer	FHO	Flying	Foraging	2023	[218]
Flock by Leader	FL	Flying	Movement	2012	[219]
Frilled Lizard Optimization	FLO	Terrestrial	Foraging	2024	[220]
Fruit Fly Optimization Algorithm	FOA	Flying	Foraging	2012	[221]
Falcon Optimization Algorithm	FOA.2	Flying	Foraging	2019	[222]
FOX-inspired Optimization Algorithm	FOX	Terrestrial	Foraging	2023	[223]
Fish-Swarm Algorithm	FSA	Aquatic	Foraging	2002	[224]
Fish Swarm Algorithm	FSA.1	Aquatic	Foraging	2011	[225]
Fish School Search	FSS	Aquatic	Foraging	2008	[226]
Green Anaconda Optimization	GAO	Terrestrial	Foraging	2023	[227]
Giant Armadillo Optimization	GAO.1	Terrestrial	Foraging	2023	[228]
Group Escape Behavior	GEB	Aquatic	Movement	2011	[229]
Golden Eagle Optimizer	GEO	Flying	Foraging	2021	[230]
Golden Jackal Optimization Algorithm	GJO	Terrestrial	Foraging	2023	[231]
Genghis Khan Shark Optimizer	GKSO	Aquatic	Foraging	2023	[232]
Good Lattice Swarm Optimization	GLSO	Other	Movement	2007	[233]
Grasshopper Optimisation Algorithm	GOA	Terrestrial	Foraging	2017	[118]
Gazelle Optimization Algorithm	GOA.1	Terrestrial	Movement	2023	[234]
Goat Search Algorithms	GSA.2	Terrestrial	Movement	2022	[235]
Glowworm Swarm Optimization	GSO	Micro	Movement	2013	[236]
Group Search Optimizer	GSO.1	Other	Movement	2009	[237]
Goose Team Optimization	GTO	Flying	Movement	2008	[238]
Gorilla Troops Optimizer	GTO.1	Terrestrial	Movement	2021	[239]
Grey Wolf Optimizer	GWO	Terrestrial	Foraging	2014	[240]
Hitchcock Birds-Inspired Algorithm	HBIA	Flying	Movement	2020	[241]
Honey-Bees Mating Optimization Algorithm	HBMO	Flying	Movement	2006	[242]
Hunger Games Search	HGS	Other	Foraging	2021	[243]
Harry's Hawk Optimization Algorithm	HHO	Flying	Foraging	2019	[244]
Hoopoe Heuristic Optimization	HHO.1	Flying	Foraging	2012	[245]
Horned Lizard Optimization Algorithm	HLOA	Terrestrial	Movement	2024	[246]
Horse Optimization Algorithm	HOA	Terrestrial	Movement	2020	[247]
Hunting Search	HuS	Other	Foraging	2010	[248]
Honeybee Social Foraging	HSF	Flying	Foraging	2007	[249]
Hierarchical Swarm Model	HSM	Other	Movement	2010	[250]
Hammerhead Shark Optimization Algorithm	HSOA	Aquatic	Foraging	2019	[251]
Humboldt Squid Optimization Algorithm	HSOA.1	Aquatic	Foraging	2023	[252]
Hypercube Natural Aggregation Algorithm	HYNAA	Other	Movement	2019	[253]
Improved Raven Roosting Algorithm	IRRO	Flying	Movement	2018	[254]
Invasive Tumor Optimization Algorithm	ITGO	Micro	Movement	2015	[255]

Table 6: Nature- and bio-inspired meta-heuristics within the *Swarm Intelligence* category (IV).

Swarm Intelligence (IV)

Algorithm Name	Acronym	Subcategory	Type	Year	Reference
Jaguar Algorithm	JA	Terrestrial	Foraging	2015	[256]
Jellyfish Search	JS	Aquatic	Movement	2021	[257]
Japanese Tree Frogs Calling Algorithm	JTFCA	Terrestrial	Movement	2012	[258]
Krill Herd	KH	Aquatic	Foraging	2012	[259]
Kookaburra Optimization Algorithm	KOA	Flying	Foraging	2023	[260]
Krestrel Search Algorithm	KSA	Flying	Foraging	2016	[261]
Killer Whale Algorithm	KWA	Aquatic	Foraging	2017	[262]
Lion Algorithm	LA	Terrestrial	Foraging	2012	[263]
Seven-Spot Labybird Optimization	LBO	Flying	Foraging	2013	[264]
Lyrebird Optimization Algorithm	LBO.1	Flying	Movement	2023	[265]
Laying Chicken Algorithm	LCA	Terrestrial	Movement	2017	[266]
Lion Optimization Algorithm	LOA	Terrestrial	Foraging	2016	[267]
Lion Pride Optimizer	LPO	Terrestrial	Movement	2012	[268]
Locust Swarms Optimization	LSO	Aquatic	Foraging	2009	[269]
Leopard Seal Optimization	LSO.1	Terrestrial	Foraging	2023	[270]
Locust Swarms Search	LSS	Aquatic	Movement	2015	[271]
Mayfly Optimization Algorithm	MA.1	Flying	Movement	2020	[272]
Magnetotactic Bacteria Optimization Algorithm	MBO	Micro	Movement	2013	[273]
Monarch Butterfly Optimization	MBO.1	Flying	Movement	2017	[274]
Migrating Birds Optimization	MBO.2	Flying	Movement	2012	[275]
Mouth Breeding Fish Algorithm	MBF	Aquatic	Foraging	2018	[276]
Migration-Crossover Algorithm	MCA	Other	Movement	2024	[277]
Modified Cuckoo Search	MCS	Flying	Foraging	2009	[278]
Modified Cockroach Swarm Optimization	MCSO	Terrestrial	Foraging	2011	[279]
Moth Flame Optimization Algorithm	MFO	Flying	Movement	2015	[280]
Mosquito Flying Optimization	MFO.1	Flying	Foraging	2016	[281]
Meerkats Inspired Algorithm	MIA	Terrestrial	Movement	2018	[282]
Mycorrhiza Optimization Algorithm	MOA	Micro	Movement	2023	[283]
Mox Optimization Algorithm	MOX	Flying	Movement	2011	[284]
Marine Predators Algorithm	MPA	Aquatic	Foraging	2020	[285]
Monkey Search	MS	Terrestrial	Foraging	2007	[286]
Moth Search Algorithm	MS.2	Flying	Movement	2018	[287]
Mantis Search Algorithm	MSA	Terrestrial	Foraging	2023	[288]
Natural Aggregation Algorithm	NAA	Other	Movement	2016	[289]
Naked Moled Rat	NMR	Terrestrial	Movement	2019	[290]
Nutcracker Optimization Algorithm	NOA	Flying	Movement	2023	[291]
Nomadic People Optimizer	NPO	Other	Movement	2019	[292]
Orcas Intelligence Algorithm	OA	Aquatic	Foraging	2020	[293]
OptBees	OB	Flying	Foraging	2012	[294]
Optimal Foraging Algorithm	OFA	Other	Foraging	2017	[295]
Owls Optimization Algorithm	OOA	Flying	Movement	2019	[296]
Osprey Optimization Algorithm	OOA.1	Flying	Foraging	2023	[297]
Orca Predation Algorithm	OPA	Aquatic	Foraging	2022	[298]
Pity Beetle Algorithm	PBA	Terrestrial	Foraging	2018	[299]
Polar Bear Optimization Algorithm	PBOA	Terrestrial	Foraging	2017	[300]
Physarum-inspired Competition Algorithm	PCA.1	Micro	Movement	2023	[301]
Prairie Dog Optimization Algorithm	PDO	Terrestrial	Foraging	2022	[302]

Table 7: Nature- and bio-inspired meta-heuristics within the *Swarm Intelligence* category (V).

Swarm Intelligence (V)

Algorithm Name	Acronym	Subcategory	Type	Year	Reference
Pigeon Inspired Optimization	PIO	Flying	Movement	2014	[303]
Population Migration Algorithm	PMA	Other	Movement	2009	[304]
Puma Optimizer	PO.1	Terrestrial	Movement	2024	[305]
Pelican Optimization Algorithm	POA.1	Flying	Foraging	2022	[306]
Pufferfish Optimization Algorithm	POA.2	Aquatic	Movement	2024	[307]
Prey Predator Algorithm	PPA	Other	Foraging	2015	[308]
Particle Swarm Optimization	PSO	Flying	Movement	1995	[80]
Penguins Search Optimization Algorithm	PSOA	Aquatic	Foraging	2013	[309]
Regular Butterfly Optimization Algorithm	RBOA	Flying	Foraging	2019	[310]
Red Deer Algorithm	RDA	Terrestrial	Movement	2016	[311]
Red Fox Optimization Algorithm	RFO	Terrestrial	Foraging	2021	[312]
Rhino Herd Behavior	RHB	Terrestrial	Movement	2018	[313]
Rock Hyraxes Swarm Optimization	RHSO	Terrestrial	Foraging	2021	[314]
Roach Infestation Problem	RIO	Terrestrial	Foraging	2008	[315]
Raccoon Optimization Algorithm	ROA	Terrestrial	Foraging	2018	[316]
Reincarnation Concept Optimization Algorithm	ROA.1	Other	Movement	2010	[317]
Red Piranha Optimization	RPO	Aquatic	Foraging	2023	[318]
Red Panda Optimization Algorithm	RPO.1	Terrestrial	Foraging	2023	[319]
Raven Roosting Optimization Algorithm	RROA	Flying	Foraging	2015	[320]
Red-tailed Hawk Algorithm	RTH	Flying	Foraging	2023	[321]
Reptile Search Algorithm	RSA	Terrestrial	Foraging	2022	[322]
Rat Swarm Optimizer	RSO	Terrestrial	Foraging	2021	[323]
Ringed Seal Search	RSS	Aquatic	Movement	2015	[324]
Shark Search Algorithm	SA	Aquatic	Foraging	1998	[325]
Swarm Bipolar Algorithm	SBA	Other	Movement	2024	[326]
Simulated Bee Colony	SBC	Flying	Foraging	2009	[327]
Satin Bowerbird Optimizer	SBO	Flying	Movement	2017	[328]
Sine Cosine Algorithm	SCA.2	Other	Movement	2016	[329]
Sand Cat Swarm Optimization	SCSO	Terrestrial	Movement	2023	[330]
Snap-Drift Cuckoo Search	SDCS	Flying	Foraging	2016	[331]
Shuffled Frog-Leaping Algorithm	SFLA	Terrestrial	Movement	2006	[332]
Spotted Hyena Optimizer	SHO	Terrestrial	Foraging	2017	[333]
Selfish Herds Optimizer	SHO.1	Terrestrial	Movement	2017	[334]
Sea-horse Optimizer	SHO.2	Aquatic	Movement	2023	[335]
Swarm Inspired Projection Algorithm	SIP	Flying	Foraging	2009	[336]
Slime Mould Algorithm	SMA	Micro	Foraging	2008	[337]
Sperm Motility Algorithm	SMA.1	Other	Movement	2017	[338]
Spider Monkey Optimization	SMO	Terrestrial	Foraging	2014	[339]
Starling Murmuration Optimizer	SMO.1	Flying	Movement	2022	[340]
Seeker Optimization Algorithm	SOA	Other	Movement	2007	[341]
Seagull Optimization Algorithm	SOA.1	Flying	Foraging	2019	[342]
Sandpiper Optimization Algorithm	SOA.2	Flying	Foraging	2020	[343]
Sailfish Optimizer Algorithm	SOA.3	Aquatic	Foraging	2019	[344]
Serval Optimization Algorithm	SOA.4	Terrestrial	Foraging	2022	[345]
Symbiosis Organisms Search	SOS	Other	Movement	2014	[346]
Sooty Tern Optimization Algorithm	STOA	Flying	Movement	2019	[347]
Social Spider Algorithm	SSA	Terrestrial	Foraging	2015	[348]

Table 8: Nature- and bio-inspired meta-heuristics within the *Swarm Intelligence* category (VI).

Swarm Intelligence (VI)

Algorithm Name	Acronym	Subcategory	Type	Year	Reference
Squirrel Search Algorithm	SSA.1	Flying	Movement	2019	[349]
Salp Swarm Algorithm	SSA.2	Aquatic	Foraging	2017	[350]
Sparrow Search Algorithm	SSA.3	Flying	Foraging	2020	[351]
Sling-shot Spider Optimization	S^2SO	Terrestrial	Foraging	2023	[352]
Shark Smell Optimization	SSO	Aquatic	Foraging	2016	[353]
Swallow Swarm Optimization	SSO.1	Flying	Foraging	2013	[354]
Social Spider Optimization	SSO.2	Terrestrial	Foraging	2013	[355]
Sperm Swarm Optimization Algorithm	SSOA	Other	Movement	2018	[356]
See-See Partidge Chicks Optimization	SSPCO	Flying	Movement	2015	[357]
Surface-Simplex Swarm Evolution Algorithm	SSSE	Other	Movement	2017	[358]
Siberian Tiger Optimization	STO	Terrestrial	Foraging	2022	[359]
Sperm Whale Algorithm	SWA	Aquatic	Movement	2016	[360]
Spider Wasp Optimizer	SWO.1	Terrestrial	Foraging	2023	[361]
Termite Hill Algorithm	TA	Terrestrial	Foraging	2012	[362]
Termite Alate Optimization Algorithm	TAOA	Terrestrial	Movement	2023	[363]
Termite Colony Optimization	TCO	Terrestrial	Foraging	2010	[364]
Tasmanian Devil Optimization	TDO	Terrestrial	Foraging	2022	[365]
Tomtit Flock Optimization Algorithm	TFOA	Flying	Foraging	2022	[366]
The Great Salmon Run Algorithm	TGSR	Aquatic	Movement	2013	[367]
Termite Life Cycle Optimizer	TLCO	Terrestrial	Movement	2023	[368]
Tyrannosaurus Optimization Algorithm	TROA	Terrestrial	Foraging	2023	[369]
Tunicate Swarm Algorithm	TSA.1	Micro	Foraging	2020	[370]
Tangent Search Algorithm	TSA.2	Other	Movement	2022	[371]
Virtual Ants Algorithm	VAA	Flying	Foraging	2006	[372]
Virtual Bees Algorithm	VBA	Flying	Foraging	2005	[373]
Virus Colony Search	VCS	Micro	Movement	2016	[374]
Virus Optimization Algorithm	VOA.1	Micro	Movement	2009	[375]
Viral Systems Optimization	VSO	Micro	Movement	2008	[376]
Wasp Colonies Algorithm	WCA	Flying	Foraging	1991	[377]
Wolf Colony Algorithm	WCA.1	Terrestrial	Foraging	2011	[378]
Worm Optimization	WO	Micro	Foraging	2014	[379]
Whale Optimization Algorithm	WOA	Aquatic	Foraging	2016	[380]
Walruses Optimization Algorithm	WaOA	Terrestrial	Movement	2023	[381]
Wolf Pack Search	WPS	Terrestrial	Foraging	2007	[382]
Weightless Swarm Algorithm	WSA	Other	Movement	2012	[383]
Wolf Search Algorithm	WSA.1	Terrestrial	Foraging	2012	[384]
Wasp Swarm Optimization	WSO	Flying	Foraging	2005	[385]
White Shark Optimizer	WSO.1	Aquatic	Foraging	2022	[386]
Yellow Saddle Goldfish	YSGA	Aquatic	Foraging	2018	[387]
Zebra Optimization Algorithm	ZOA	Terrestrial	Foraging	2022	[388]
Zombie Survival Optimization	ZSO	Other	Foraging	2012	[389]

- **Movement:** We have considered that an algorithm belongs to the *movement inspiration* subcategory if the biological inspiration resides mainly in the way the animal inspiring the algorithm regularly moves around its environment. As such, the differential aspect of the movement could hinge on the dynamics of the movement itself (e.g. the flying movement of birds in PSO [80], jumping actions as in Shuffled Frog-Leaping Algorithm, SFLA [332], or by aquatic movements as in DPO [201]), or by the movement of the population (correspondingly, swarming movements as in Bird Swarm Algorithm, BSA [162], the migration of populations like Population Migration Algorithm, PMA [304], or the migration of particular animals like salmon [367], among others).

- **Foraging:** Rather than the movement strategy, in some other algorithmic variants it is the mechanism used to obtain their food what drives the behavior of the animal and, ultimately, the design of the meta-heuristic algorithm. This foraging behavior can in turn be observed in many flavors, from the tactics used by the animal at hand to surround its food source (as in the aforementioned GWO [240] and LA [263]), inspired in breeding nutrition (as Cuckoo Search [188, 390]), inspired in hunting techniques observed in grey wolves and lions, respectively), particular mechanisms to locate food sources and communicate its existence to the rest of the swarm (as in ACO [115]), or other exploration strategies such as the echolocation in dolphins [195], or the flashing attraction between partners observed in fireflies [117]. Sometimes, the movement of the animal also obeys to food search and retrieval. In this case, we consider that the algorithm belongs to the *foraging* inspiration type, rather than to the movement type. Nowadays, inspiration by foraging mechanisms is becoming more and more consolidated in the research community, appearing explicitly in the name of several bio-inspired algorithms.

3.3 Physics/Chemistry based Algorithms

Algorithms under this category are characterized by the fact that they imitate the behavior of physical or chemical phenomena, such as gravitational forces, electromagnetism, electric charges and water movement (in relation to physics-based approaches), and chemical reactions and gases particles movement as for chemistry-based optimization algorithms.

The complete list of reviewed algorithms in this category is provided in Tables 9 and 10 (physics-based algorithms) and Table 11 (chemistry-based methods). In this category we can find some well-known algorithms reported in the last century such as Simulated Annealing [79], or one of the most important algorithms in physics-based meta-heuristic optimization, Gravitational Search Algorithm, GSA [391]. Interestingly, a variety of space-based algorithms are rooted in GSA, such as Black Hole optimization (BH, [392]) or Galaxy Based Search Algorithm (GBSA, [393]). Other algorithms such as Harmony Search (HS, [394]) relate to the music composition process, a human invention that has more in common with other physical algorithms in what refers to the usage of sound waves than with Social Human Behavior based algorithms, the category discussed in what follows.

3.4 Social Human Behavior based Algorithms

Algorithms falling in this category are inspired by human social concepts, such as decision-making and ideas related to the expansion/competition of ideologies inside the society as ideology (Ideology Algorithm, IA, [466]), or political concepts such as the Imperialist Colony Algorithm (ICA, [467]). This category also includes algorithms that emulate sports competitions such as the Soccer League Competition Algorithm (SLC, [468]). Brainstorming processes have also laid the inspirational foundations of several algorithms such as Brain Storm Optimization algorithm (BSO.2, [469]) and Global-Best Brain Storm Optimization algorithm (GBSO, [470]). The complete list of algorithms in this category is given in Table 12 and in Table 13.

3.5 Plants based Algorithms

This category essentially gathers all optimization algorithms whose search process is inspired by plants. In this case, as opposed to other methods within the Swarm Intelligence category, there is no communication between agents. One of the most well-known is Forest Optimization Algorithms (FOA.1, [523]), inspired by the process of plant reproduction. Table 14 details the specific algorithms classified in this category.

3.6 Algorithms with Miscellaneous Sources of Inspiration

In this category there are included the algorithms that do not fit in any of the previous categories, i.e., we can find algorithms of diverse characteristics such as the Ying-Yang Pair Optimization (YYOP, [546]). Although this defined category

Table 9: Nature- and bio-inspired meta-heuristics within the *Physics based* category (I).

Physics based (I)

Algorithm Name	Acronym	Year	Reference
Artificial Electric Field Algorithm	AEFA	2019	[395]
Archimedes Optimization Algorithm	AOA.1	2021	[396]
Artificial Physics Optimization	APO	2009	[397]
Atom Search Optimization	ASO.1	2019	[398]
Big Bang Big Crunch	BBBC	2006	[399]
Black Hole Optimization	BH	2013	[392]
Colliding Bodies Optimization	CBO	2014	[400]
Crystal Energy Optimization Algorithm	CEO	2016	[401]
Central Force Optimization	CFO	2008	[402]
Charged Systems Search	CSS	2010	[403]
Electromagnetic Field Optimization	EFO	2016	[404]
Electromagnetism Mechanism Optimization	EMO	2003	[405]
Electimize Optimization Algorithm	EOA.1	2011	[406]
Electron Radar Search Algorithm	ERSA	2020	[407]
Galaxy Based Search Algorithm	GBSA	2011	[393]
Gravitational Clustering Algorithm	GCA	1999	[408]
Gravitational Emulation Local Search	GELS	2009	[409]
Gravitational Field Algorithm	GFA	2010	[410]
Geyser Inspired Algorithm	GIA	2023	[411]
Gravitational Interactions Algorithm	GIO	2011	[412]
General Relativity Search Algorithm	GRSA	2015	[413]
Gravitational Search Algorithm	GSA	2009	[391]
Galactic Swarm Optimization	GSO.2	2016	[414]
Hydrological Cycle Algorithm	HCA	2017	[415]
Harmony Elements Algorithm	HEA	2009	[416]
Hysteresis for Optimization	HO	2002	[417]
Hurricane Based Optimization Algorithm	HO.2	2014	[418]
Harmony Search	HS	2005	[394]
Intelligent Gravitational Search Algorithm	IGSA	2012	[419]
Intelligence Water Drops Algorithm	IWD	2009	[420]
Light Ray Optimization	LRO	2010	[421]
Lightning Search Algorithm	LSA	2015	[422]
Magnetic Optimization Algorithm	MFO.2	2008	[423]
Method of Musical Composition	MMC	2014	[424]
Melody Search	MS.1	2011	[425]
Multi-Verse Optimizer	MVO	2016	[426]
Newton-Raphson-Based Optimizer	NRBO	2024	[427]
Optics Inspired Optimization	OIO	2015	[428]
Particle Collision Algorithm	PCA	2007	[429]
PopMusic Algorithm	PopMusic	2002	[430]
Quantum Superposition Algorithm	QSA	2015	[431]
Rain-Fall Optimization Algorithm	RFOA	2017	[432]
Rain Water Algorithm	RWA	2017	[433]
River Formation Dynamics	RFD	2007	[434]
Radial Movement Optimization	RMO	2014	[435]
Ray Optimization	RO	2012	[436]

Table 10: Nature- and bio-inspired meta-heuristics within the *Physics based* category (II).

Physics based (II)			
Algorithm Name	Acronym	Year	Reference
Snow Ablation Optimizer	SAO	2023	[437]
Space Gravitational Algorithm	SGA	2005	[438]
Sonar Inspired Optimization	SIO	2017	[439]
States Matter Optimization Algorithm	SMS	2014	[440]
Spiral Dynamics Optimization	SO	2011	[441]
Spiral Optimization Algorithm	SPOA	2010	[442]
Self-Driven Particles	SPP	1995	[443]
Solar System Algorithm	SSA.4	2021	[444]
Turbulent Flow of Water-based Optimization	TFWO	2020	[445]
Vibrating Particle Systems Algorithm	VPO	2017	[446]
Vortex Search Algorithm	VS	2015	[447]
Water Cycle Algorithm	WCA.2	2012	[448]
Water Evaporation Optimization	WEO	2016	[449]
Water Flow-Like Algorithms	WFA	2007	[450]
Water Flow Algorithm	WFA.1	2007	[451]
Water-Flow Algorithm Optimization	WFO	2011	[452]
Water Wave Optimization Algorithm	WWA	2015	[453]

Table 11: Nature- and bio-inspired meta-heuristics within the *Chemistry based* category.

Chemistry based			
Algorithm Name	Acronym	Year	Reference
Artificial Chemical Process	ACP	2005	[454]
Artificial Chemical Reaction Optimization Algorithm	ACROA	2011	[455]
Artificial Reaction Algorithm	ARA	2013	[456]
Chemical Reaction Optimization Algorithm	CRO.1	2010	[457]
Gases Brownian Motion Optimization	GBMO	2013	[458]
Heat Transfer Search Algorithm	HTS	2015	[459]
Ions Motion Optimization Algorithm	IMO	2015	[460]
Integrated Radiation Optimization	IRO	2007	[461]
Kinetic Gas Molecules Optimization	KGMO	2014	[462]
Photosynthetic Algorithm	PA	1999	[463]
Simulated Annealing	SA.1	1989	[79]
Synergistic Fibroblast Optimization	SFO	2017	[464]
Thermal Exchange Optimization	TEO	2017	[465]

Table 12: Nature- and bio-inspired meta-heuristics within the *Social Human Behavior* based category.

Social Human Behavior (I)

Algorithm Name	Acronym	Year	Reference
Adolescent Identity Search Algorithm	AISA	2020	[471]
Anarchic Society Optimization	ASO	2012	[472]
Alpine Skiing Optimization	ASO.2	2022	[473]
Brain Storm Optimization Algorithm	BSO.2	2011	[469]
Bus Transportation Behavior	BTA	2019	[474]
Collective Decision Optimization Algorithm	CDOA	2017	[475]
Cognitive Behavior Optimization Algorithm	COA.3	2016	[476]
Competitive Optimization Algorithm	COOA	2016	[477]
Community of Scientist Optimization Algorithm	CSOA	2012	[478]
Cultural Algorithms	CA	1999	[479]
Duelist Optimization Algorithm	DOA	2016	[480]
Election Algorithm	EA	2015	[481]
Football Game Inspired Algorithms	FCA.1	2009	[482]
FIFA World Cup Competitions	FIFAAO	2016	[483]
Golden Ball Algorithm	GBA	2014	[484]
Global-Best Brain Storm Optimization Algorithm	GBSO	2017	[470]
Group Counseling Optimization	GCO	2010	[485]
Group Leaders Optimization Algorithm	GLOA	2011	[486]
Greedy Politics Optimization Algorithm	GPO	2014	[487]
Gaining-sharing Knowledge	GSK	2023	[488]
Group Teaching Optimization Algorithm	GTOA	2020	[489]
Human Evolutionary Model	HEM	2007	[490]
Human Group Formation	HGF	2010	[491]
Human-Inspired Algorithms	HIA	2009	[492]
Human Urbanization Algorithm	HUA	2020	[493]
Ideology Algorithm	IA	2016	[466]
Imperialist Competitive Algorithm	ICA	2007	[467]
Kho-Kho optimization Algorithm	KKOA	2020	[494]
League Championship Algorithm	LCA.1	2014	[495]
Life Choice Based Optimizer	LCBO	2020	[496]
Leaders and Followers Algorithm	LFA	2015	[497]
Old Bachelor Acceptance	OBA	1995	[498]
Oriented Search Algorithm	OSA	2008	[499]
Political Optimizer	PO	2020	[500]
Parliamentary Optimization Algorithm	POA	2008	[501]
Poor and Rich Optimization Algorithm	PRO	2019	[502]
Queuing Search Algorithm	QSA.1	2018	[503]
Search and Rescue Algorithm	SAR	2019	[504]
Social Behavior Optimization Algorithm	SBO.1	2003	[505]
Social Cognitive Optimization	SCO	2002	[506]
Social Cognitive Optimization Algorithm	SCOA	2010	[507]
Social Emotional Optimization Algorithm	SEA	2010	[508]
Stock Exchange Trading Optimization	SETO	2022	[509]
Stochastic Focusing Search	SFS	2008	[510]
Soccer Game Optimization	SGO	2012	[511]
Soccer League Competition	SLC	2014	[468]
Student Psychology Optimization Algorithm	SPBO	2020	[512]

Table 13: Nature- and bio-inspired meta-heuristics within the *Social Human Behavior* based category.

Social Human Behavior (II)			
Algorithm Name	Acronym	Year	Reference
Stadium Spectators Optimizer	SSO.3	2024	[513]
Tiki-Taka Algorithm	TTA	2020	[514]
Team Game Algorithm	TGA	2018	[515]
Teaching-Learning Based Optimization Algorithm	TLBO	2011	[516]
Thieves and Police Optimization Algorithm	TPOA	2021	[517]
Tactical Unit Algorithm	TUA	2024	[518]
Tug Of War Optimization	TWO	2016	[519]
Unconscious Search	US	2012	[520]
Volleyball Premier League Algorithm	VPL	2017	[521]
Wisdom of Artificial Crowds	WAC	2011	[522]

Table 14: Nature- and bio-inspired meta-heuristics within the *Plants based* category.

Plants based			
Algorithm Name	Acronym	Year	Reference
Artificial Flora Optimization Algorithm	AFO	2018	[524]
Artificial Plants Optimization Algorithm	APO.1	2013	[525]
BrunsVigia Flower Optimization Algorithm	BVOA	2018	[526]
Carnivorous Plant Algorithm	CPA	2021	[527]
Discrete Mother Tree Optimization	DMTO	2020	[528]
Forest Optimization Algorithm	FOA.1	2014	[523]
Flower Pollination Algorithm	FPA	2012	[529]
Lotus Effect Algorithm	LEA	2023	[530]
Natural Forest Regeneration Algorithm	NFR	2016	[531]
Plant Growth Optimization	PGO	2008	[532]
Plant Propagation Algorithm	PPA.1	2009	[533]
Paddy Field Algorithm	PFA	2009	[534]
Root Growth Optimizer	RGO	2015	[535]
Root Tree Optimization Algorithm	RTOA	2016	[536]
Runner Root Algorithm	RRA	2015	[537]
Saplings Growing Up Algorithm	SGA.1	2007	[538]
Self-Defense Mechanism Of The Plants Algorithm	SDMA	2018	[539]
Seasons Optimization	SO.1	2022	[540]
Strawberry Plant Algorithm	SPA	2014	[541]
Smart Root Search	SRS	2020	[542]
Tree Growth Algorithm	TGA.1	2019	[543]
Tree Physiology Optimization	TPO	2018	[544]
Tree Seed Algorithm	TSA	2015	[545]

is heterogeneous and does not exhibit any uniformity among the algorithms it represents, its inclusion in the taxonomy serves as an exemplifying fact of the very different sources of inspiration existing in the literature. The ultimate goal of reflecting this miscellaneous set of algorithms is to spawn new categories once more algorithms are created by recreating similar inspirational concepts that the assorted ones already present in this category.

The complete list of algorithms in this category is in Tables 15 and 16. In this regard, we stress this pressing need for grouping assorted algorithms in years to come to give rise to new categories. Otherwise, if we just stockpile new algorithms without a clear correspondence to the aforementioned categories in this miscellaneous group, the overall taxonomy will not evolve and will eventually lack its main purpose: to systematically sort and ease the analysis of future advances and achievements in the field.

Table 15: Nature- and bio-inspired meta-heuristics within the *Miscellaneous* category.

Miscellaneous (II)

Algorithm Name	Acronym	Year	Reference
Scientifics Algorithms	SA.2	2014	[547]
Social Engineering Optimization	SEO	2017	[548]
Stochastic Fractal Search	SFS.1	2015	[549]
Snow Flake Optimization Algorithm	SFO.1	2023	[550]
Search Group Algorithm	SGA.2	2015	[551]
Simple Optimization	SOPT	2012	[552]
Ship Rescue Optimization	SRO	2024	[553]
Small World Optimization	SWO	2006	[554]
The Great Deluge Algorithm	TGD	1993	[555]
Wind Driven Optimization	WDO	2010	[556]
Ying-Yang Pair Optimization	YYOP	2016	[546]

Table 16: Nature- and bio-inspired meta-heuristics within the *Miscellaneous* category.

Miscellaneous (I)

Algorithm Name	Acronym	Year	Reference
Atmosphere Clouds Model	ACM	2013	[557]
Artificial Cooperative Search	ACS	2012	[558]
Innovative Gunner Algorithm	AIG	2019	[559]
Across Neighbourhood Search	ANS	2016	[560]
Botox Optimization Algorithm	BOA.2	2024	[561]
Battle Royale Optimization Algorithm	BRO	2020	[562]
Bar Systems	BS.2	2008	[563]
Backtracking Search Optimization	BSO.3	2012	[564]
Cloud Model-Based Algorithm	CMBDE	2012	[565]
Chaos Optimization Algorithm	COA.4	1998	[566]
Clonal Selection Algorithm	CSA.1	2000	[567]
COVID-19 Optimizer Algorithm	CVA	2020	[568]
Dice Game Optimizer	DGO	2019	[569]
Dialectic Search	DS	2009	[570]
Differential Search Algorithm	DSA	2012	[571]
Exchange Market Algorithm	EMA	2014	[572]
Extremal Optimization	EO	2000	[573]
Equilibrium Optimizer	EO.1	2020	[574]
Fireworks Algorithm Optimization	FAO	2010	[575]
Farmland Fertility Algorithm	FFA	2018	[576]
Grenade Explosion Method	GEM	2010	[577]
Golden Sine Algorithm	GSA.1	2017	[578]
Golf Sport Inspired Search	GSIS	2024	[579]
Heart Optimization	HO.1	2014	[580]
Hyper-parameter Dialectic Search	HDS	2020	[581]
Ideological Sublations	IS	2017	[582]
Interior Search Algorithm	ISA	2014	[583]
Keshtel Algorithm	KA	2014	[584]
Kidney-Inspired Algorithm	KA.1	2017	[585]
Kaizen Programming	KP	2014	[586]
Liver Cancer Algorithm	LCA.2	2023	[587]
Literature Research Optimizer	LRO.1	2024	[588]
Membrane Algorithms	MA	2005	[589]
Mine Blast Algorithm	MBA	2013	[590]
Neuronal Communication Algorithm	NCA	2017	[591]
Nizar Optimization Algorithm	NOA.1	2024	[592]
Plasma Generation Optimization	PGO.1	2020	[593]
Pearl Hunting Algorithm	PHA	2012	[594]
Passing Vehicle Search	PVS	2016	[595]
Artificial Raindrop Algorithm	RDA.1	2014	[596]
Reactive Dialectic Search	RDS	2017	[597]

4 Taxonomy by Behavior for Population based Nature- and Bio-inspired Optimization

We now proceed with our second proposed taxonomy for population-based metaheuristics. In this case, we sort the different algorithmic proposals reported by the community by their behavior, without any regard to their source of inspiration. To this end, a clear sorting criterion is needed that, while keeping itself agnostic with respect to its inspiration, could summarize as much as possible the different behavioral procedures characterizing the algorithms under review. The criterion adopted for this purpose is the mechanisms used for creating new solutions, or for changing existing solutions to the optimization problem. These are the main features that define the search process of each algorithm.

First, we have divided the reviewed optimization algorithms into two categories:

- **Differential Vector Movement**, in which new solutions are produced by a shift or a mutation performed onto a previous solution. The newly generated solution could compete against previous ones, or against other solutions in the population to achieve a space and remain therein in subsequent search iterations. This solution generation scheme implies selecting a solution as the reference, which is changed to explore the space of variables and, effectively, produce the search for the solution to the problem at hand. The most representative method of this category is arguably PSO [80], in which each solution evolves with a velocity vector to explore the search domain. Another popular algorithm with differential movement at its core is DE [59], in which new solutions are produced by adding differential vectors to existing solutions in the population. Once a solution is selected as the reference one, it is perturbed by adding the difference between other solutions. The decision as to which solutions from the population are influential in the movement is a decision that has an enormous influence on the behavior of the overall search. Consequently, we further divide this category by that decision. The movement – thus, the search – can be guided by i) all the population (Figure 4.a); ii) only the significant/relevant solutions, e.g., the best and/or the worst candidates in the population (Figure 4.b); or iii) a small group, which could stand for the neighborhood around each solution or, in algorithms with subpopulations, only the subpopulation to which each solution belongs (Figure 4.c).
- **Solution creation**, in which new solutions are not generated by mutation/movement of a single reference solution, but instead by combining several solutions (so there is not only a single parent solution), or other similar mechanism. Two approaches can be utilized for creating new solutions. The first one is by combination, or crossover of several solutions (Figure 4.d). The classical GA [98] is the most straightforward example of this type. Another approach is by stigmergy (Figure 4.e), in which there is indirect coordination between the different solutions or agents, usually using an intermediate structure, to generate better ones. A classical example of stigmergy for creating solutions is ACO [598], in which new solutions are generated by the trace of pheromones left by different agents on a graph representing the solution space of the problem under analysis.

Bearing the above criteria in mind, Figure 5 shows the classification reached after our literature analysis. The plot indicates, for the 518 reviewed algorithms, the number and ratio of proposals classified in each category and subcategory. It can be observed that in most nature- and bio-inspired algorithms, new solutions are generated by differential vector movement over existing ones (69% vs 31%). Among them, the search process is mainly guided by representative solutions (almost 60% in global, 86% from this category), mainly the so-called current best solution (in a very similar fashion to the naive version of the PSO solver). Thus, the creation of new solutions by movement vectors oriented towards the best solution is the search mechanism found in more than half (almost 60%) of all the 518 reviewed proposals.

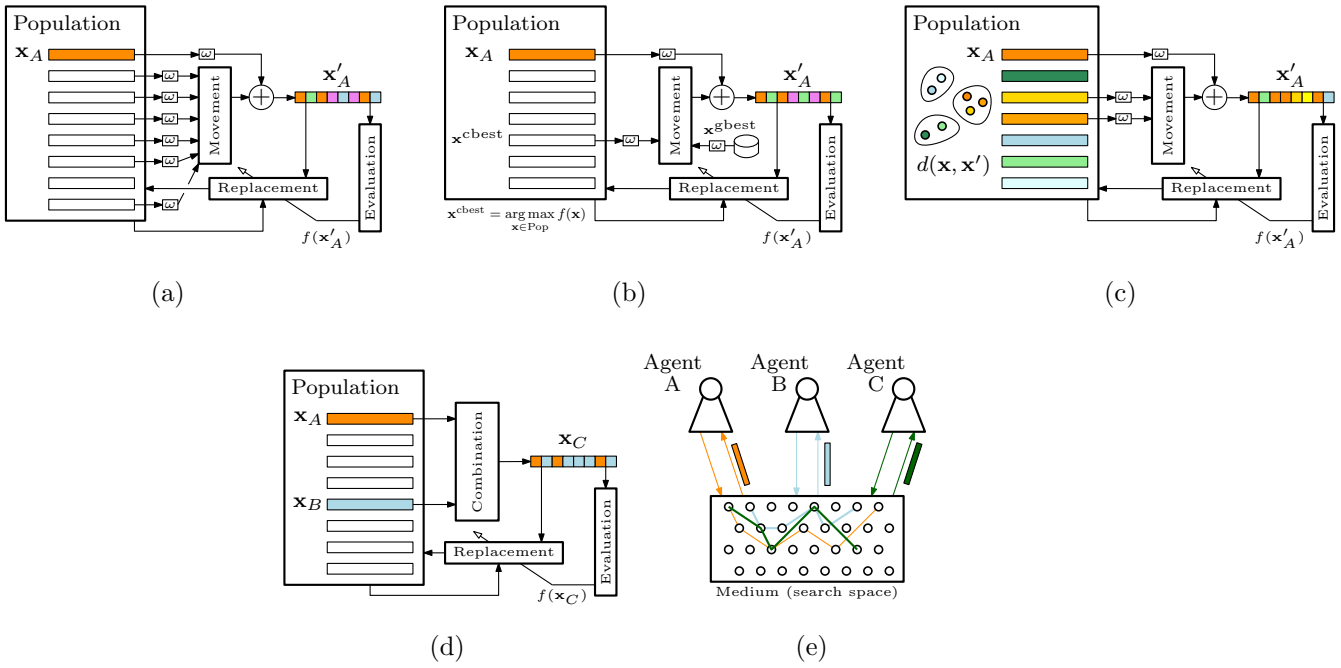


Figure 4: Schematic diagrams of the different algorithmic behaviors on which our second taxonomy relies. The upper plots illustrate the process of generating new solutions by *Differential Vector Movement* from a given solution $\mathbf{x}_A$, using (a) the entire population; (b) relevant individuals (in the example, the movement results from a weighted combination – ω– of the current best solution in the population and the best solution found so far by the algorithm); and (c) neighboring solutions in the population to the reference individual. The lower plots show the same process using *solution creation* by (d) combination; and (e) stigmergy.

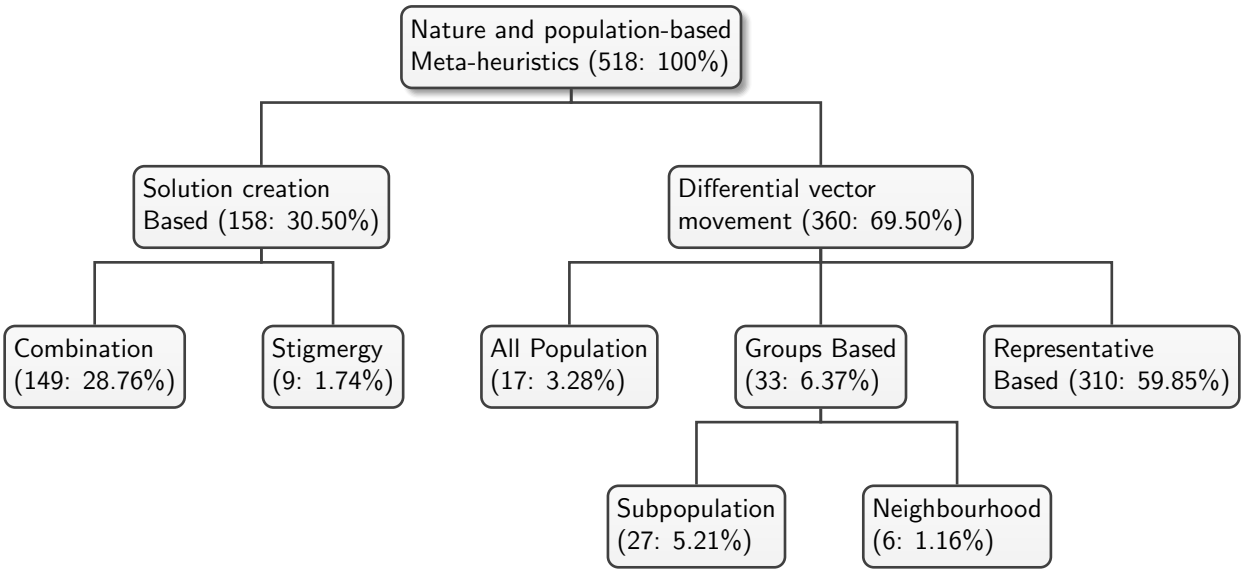


Figure 5: Classification of the reviewed papers using the *behavior* taxonomy.

The following subsections provide a brief global view of the different categories introduced above. For each category, we describe its main characteristics, an example, and a table with the algorithms belonging to that category.

4.1 Differential Vector Movement

This category of our behavior-based taxonomy amounts up to 69% of the analyzed algorithms. In all of them, new solutions are obtained by a movement departing from existing solutions. By using a solution as the reference, a differential vector is used to *move* from the reference towards a new candidate, that could replace the previous one or instead compete to be included into the population.

The crucial decision in differential vector movement is how the differential vector (namely, the intensity and direction of the movement) is calculated. This differential vector could be calculated so as to move the reference solution to another solution (usually a better one), or as a lineal combination of other different solutions, allowing the combination of attraction vectors (toward the best solutions) with repulsion vectors (away from worse ones, or from other solutions, to enforce diversity). The mathematical nature of this operation usually restricts the domain of the representation to a numerical, usually real-valued representation.

This category is further divided into subcategories as a function of the above decision, i.e. which solutions are considered to create the movement vector. It should be noted that some algorithms can be classified into more than one subcategory. For instance, a particle's update in the PSO solver is affected by the global best particle behavior and certain local best particle(s) behavior. The local best behavior can be either dependent on the particle's previous behavior or the behavior of some particles in its neighborhood. This makes PSO a possible member of two of the subcategories, namely, *Differential Vector as a Function of Representative Solutions* and *Differential Vector as a Function of a Group of Solutions*. Nevertheless, we have considered the classical PSO as a member of *Representative Solutions* because the influence of the best algorithm is stronger than the influence of the neighborhood. In any case, following the above rationale, other PSO variants could fall within any other subcategory. We now describe each of such subcategories.

4.1.1 Differential Vector as a Function of the Entire Population

One possible criterion is to use all the individuals in the population to generate the movement of each solution. In these algorithms, all individuals have a degree of influence on the movement of the other solutions. Such a degree is usually weighted according to the fitness difference and/or distance between solutions. A significant example is FA [117], in which a solution suffers a moving force towards better solutions as a function of their distance. Consequently, solutions closer to the reference solution will have a stronger influence than more distant counterparts. As shown in Table 17, algorithms in this subcategory belong to different categories in the previous *inspiration source* based taxonomy.

4.1.2 Differential Vector as a Function of Representative Solutions

In this group (the most populated in this second taxonomy), the different movement of each solution is only influenced by a small group of representative solutions. It is often the case that these representative solutions are selected to be the best solutions found by the algorithm (as per the objective of the problem at hand), being able to be guided only by e.g. the current best individual of the population.

Tables 18, 19, 20, 21, 22, 23 and 24 show the different algorithms in this subcategory. An exemplary algorithm of this category that has been a major meta-heuristic solver in the history of the field is PSO [80]. In this solver, each solution or particle is guided by the global current best solution and the best solution obtained by that particle during the search. Another classical algorithm in this category is the majority of the family of DE approaches [59]. In most of the variants of this evolutionary algorithm, the influence of the best solution(s) is hybridized with a differential vector that perturbs the new solution toward random individuals for the sake of increased diversity along the search. However, this subcategory also includes many other algorithms with differences in considering nearly better solutions (as in the Bat Inspired Algorithm [153] or the Brain Storm Optimization Algorithm [469]) or the worse solutions (to avoid less promising regions), as in the Grasshopper Optimization Algorithm (GOA, [118]). More than half of all algorithmic proposals dwell in this subcategory, with a prominence of Swarm Intelligence solvers due to their behavioral inspiration in PSO and DE. We will revolve around these identified similarities in Section 5.

4.1.3 Differential Vector as a Function of a Group of Solutions

Algorithms within this category do not resort to representative solutions of the entire population (such as the current best), but they only consider solutions of a subset or group of the solutions in the population. When the differential movement

Table 17: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by the entire population.

Influenced by the entire population

Algorithm Name	Acronym	Year	Reference
Artificial Electric Field Algorithm	AEFA	2019	[395]
Artificial Plants Optimization Algorithm	APO.1	2013	[525]
Botox Optimization Algorithm	BOA.2	2024	[561]
Chaotic Dragonfly Algorithm	CDA	2018	[177]
Central Force Optimization	CFO	2008	[402]
Charged Systems Search	CSS	2010	[403]
Dwarf Mongoose Optimization	DMO	2022	[198]
Electromagnetism Mechanism Optimization	EMO	2003	[405]
Firefly Algorithm	FA	2009	[117]
Gravitational Clustering Algorithm	GCA	1999	[408]
Group Counseling Optimization	GCO	2010	[485]
Gravitational Search Algorithm	GSA	2009	[391]
Human Group Formation	HGF	2010	[491]
Hoopoe Heuristic Optimization	HHO.1	2012	[245]
Intelligent Gravitational Search Algorithm	IGSA	2012	[419]
Integrated Radiation Optimization	IRO	2007	[461]
Locust Swarms Search	LSS	2015	[271]

considers both a group and a representative of all the population, the algorithm under analysis is considered to belong to the previous subcategory, because the representative has usually the strongest influence over the search. Two different subcategories hold when a group of solutions is used for computing the differential movement vector:

- **Subpopulation based differential vector**: In algorithms belonging to this subcategory (listed in Table 25) the population is divided in several subpopulations, such that the movement of each solution is only affected by the other solutions in the same subpopulation. Examples of algorithms in this subcategory are LA [263] or the Monarch Butterfly Optimization algorithm (MBO, [274]).
- **Neighborhood based differential vector**: In this subcategory, each solution is affected only by solutions in its local neighborhood. Table 26 compiles all algorithms that are classified in this subcategory. A notable example in this list is BFOA [148], in which all solutions in the neighborhood impact on the computation of the movement vector, either by attracting the solution (if the neighboring solution has better fitness than the reference solution) or in a repulsive way (when the neighboring solution is worse than the one to be moved).

4.2 Solution Creation

This category is composed of algorithms that explore the domain search by generating new solutions, not by moving existing ones. This group is a significant ratio (almost 31%) of all proposals, and includes many classical algorithms like GA [98]. A very widely exploited advantage of these methods is the possibility to adapt the generation method to the particular problem, hence allowing for different possible representations and, therefore, easing its application to a wider range of problems. In the following, we describe the different subcategories that result from the diverse mechanisms by which solutions can be created.

4.2.1 Creation by Combination

The most common option to generate a new solution is to combine existing ones. In these algorithms, different solutions are selected and combined using a crossover operator or combining method to give rise to new solutions. The underlying idea

Table 18: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by representative solutions (I).

Influenced by representative solutions (I)			
Algorithm Name	Acronym	Year	Reference
Artificial Algae Algorithm	AAA	2015	[120]
Artificial Bee Colony	ABC	2007	[116]
Animal Behavior Hunting	ABH	2014	[122]
African Buffalo Optimization	ABO	2016	[123]
Atmosphere Clouds Model	ACM	2013	[557]
Artificial Ecosystem Optimizer	AEO	2020	[83]
Artificial Feeding Birds	AFB	2018	[125]
Artificial Hummingbird Algorithm	AHA	2022	[126]
Archerfish Hunting Optimizer	AHO	2022	[127]
Adolescent Identity Search Algorithm	AISA	2020	[471]
Ant Lion Optimizer	ALO	2015	[130]
Animal Migration Optimization	AMO	2014	[128]
Aphid Metaheuristic Optimization	AMO.1	2022	[129]
Across Neighbourhood Search	ANS	2016	[560]
Aquila Optimizer	AO	2021	[131]
Archimedes Optimization Algorithm	AOA.1	2021	[396]
Arithmetic Optimization Algorithm	AOA.2	2021	[133]
Artificial Rabbits Optimization	ARO.1	2022	[134]
Anarchic Society Optimization	ASO	2012	[472]
Atom Search Optimization	ASO.1	2019	[398]
Alpine Skiing Optimization	ASO.2	2022	[473]
Artificial Searching Swarm Algorithm	ASSA	2009	[135]
Artificial Tribe Algorithm	ATA	2009	[136]
African Wild Dog Algorithm	AWDA	2013	[137]
Bison Behavior Algorithm	BBA	2019	[142]
Big Bang Big Crunch	BBBC	2006	[399]
Bacterial Chemotaxis Optimization	BCO.2	2002	[146]
Bacterial Colony Optimization	BCO.1	2012	[145]
Border Collie Optimization	BCO.3	2020	[147]
Bald Eagle Search Optimization	BES	2019	[139]
Black Hole Optimization	BH	2013	[392]
Bat Intelligence	BI	2012	[152]
Bat Inspired Algorithm	BIA	2010	[153]
Biology Migration Algorithm	BMA	2019	[154]
Blind, Naked Mole-Rats Algorithm	BNMR	2013	[156]
Butterfly Optimizer	BO	2015	[157]
Bonobo Optimizer	BO.1	2019	[158]
Battle Royale Optimization Algorithm	BRO	2020	[562]
Bird Swarm Algorithm	BSA	2016	[162]
Bee Swarm Optimization	BSO	2010	[163]
Bioluminiscent Swarm Optimization Algorithm	BSO.1	2011	[164]
Brain Storm Optimization Algorithm	BSO.2	2011	[469]
Biological Survival Optimizer	BSO.4	2023	[165]
Buzzard Optimization Algorithm	BUZOA	2019	[167]
Beluga Whale Optimization	BWO.1	2022	[169]
Binary Whale Optimization Algorithm	BWOA	2019	[170]
Collective Animal Behavior	CAB	2012	[171]

Table 19: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by representative solutions (II).

Influenced by representative solutions (II)			
Algorithm Name	Acronym	Year	Reference
Catfish Optimization Algorithm	CAO	2011	[173]
Cheetah Based Algorithm	CBA	2018	[172]
Cricket Behavior-Based Algorithm	CBBE	2016	[174]
Chaotic Crow Search Algorithm	CCSA	2018	[176]
Collective Decision Optimization Algorithm	CDOA	2017	[475]
Camel Herd Algorithm	CHA	2017	[180]
Chimp Optimization Algorithm	ChOA	2020	[181]
Cloud Model-Based Algorithm	CMBDE	2012	[565]
Camel Traveling Behavior	COA.1	2016	[183]
Coyote Optimization Algorithm	COA.2	2018	[184]
Cognitive Behavior Optimization Algorithm	COA.3	2016	[476]
Chaos Optimization Algorithm	COA.4	1998	[566]
COOT Optimization Algorithm	COA.5	2021	[185]
Coati Optimization Algorithm	COA.6	2023	[186]
Competitive Optimization Algorithm	COOA	2016	[477]
Crested Porcupine Optimizer	CPO	2024	[187]
Crow Search Algorithm	CSA	2016	[189]
Chameleon Swarm Algorithm	CSA.2	2021	[81]
Circle Search Algorithm	CSA.3	2022	[190]
Cat Swarm Optimization	CSO	2006	[191]
Community of Scientist Optimization Algorithm	CSOA	2012	[478]
Dragonfly Algorithm	DA	2016	[193]
Differential Evolution	DE	1997	[93]
Dynamic Hunting Leadership	DHL	2023	[196]
Deer Hunting Optimization Algorithm	DHOA	2019	[197]
Dandelion Optimizer	DO	2022	[199]
Dingo Optimizer	DOX	2021	[200]
Dolphin Partner Optimization	DPO	2009	[201]
Differential Search Algorithm	DSA	2012	[571]
Donkey Theorem Optimization	DTO	2019	[202]
Enriched Coati Osprey Algorithm	ECOA	2024	[203]
Electric Eel Foraging Optimization	EEFO	2024	[204]
Elephant Herding Optimization	EHO	2016	[206]
Elk Herd Optimizer	EHO.1	2024	[207]
Ebola Optimization Search Algorithm	EOSA	2022	[208]
Emperor Penguin Optimizer	EPO	2018	[210]
Electron Radar Search Algorithm	ERSA	2020	[407]
Elephant Search Algorithm	ESA	2015	[212]
Eagle Strategy	ES.1	2010	[211]
Elephant Swarm Water Search Algorithm	ESWSA	2018	[213]
Fireworks Algorithm Optimization	FAO	2010	[575]
Flocking Base Algorithms	FBA	2006	[215]
Fast Bacterial Swarming Algorithm	FBSA	2008	[216]
Football Game Inspired Algorithms	FCA.1	2009	[482]
Farmland Fertility Algorithm	FFA	2018	[576]
Fire Hawk Optimizer	FHO	2023	[218]
FIFA World Cup Competitions	FIFAAO	2016	[483]

Table 20: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by representative solutions (III).

Influenced by representative solutions (III)			
Algorithm Name	Acronym	Year	Reference
Flock by Leader	FL	2012	[219]
Frilled Lizard Optimization	FLO	2024	[220]
Fruit Fly Optimization Algorithm	FOA	2012	[221]
Falcon Optimization Algorithm	FOA.2	2019	[222]
Flower Pollination Algorithm	FPA	2012	[529]
Fish-Swarm Algorithm	FSA	2002	[224]
Fish Swarm Algorithm	FSA.1	2011	[225]
Fish School Search	FSS	2008	[226]
Green Anaconda Optimization	GAO	2023	[227]
Giant Armadillo Optimization	GAO.1	2023	[228]
Gases Brownian Motion Optimization	GBMO	2013	[458]
Global-Best Brain Storm Optimization Algorithm	GBSO	2017	[470]
Group Escape Behavior	GEB	2011	[229]
Golden Eagle Optimizer	GEO	2021	[230]
Grenade Explosion Method	GEM	2010	[577]
Gravitational Field Algorithm	GFA	2010	[410]
Geyser Inspired Algorithm	GIA	2023	[411]
Gravitational Interactions Algorithm	GIO	2011	[412]
Golden Jackal Optimization Algorithm	GJO	2023	[231]
Genghis Khan Shark Optimizer	GKSO	2023	[232]
Good Lattice Swarm Optimization	GLSO	2007	[233]
Grasshopper Optimisation Algorithm	GOA	2017	[118]
Gazelle Optimization Algorithm	GOA.1	2023	[234]
General Relativity Search Algorithm	GRSA	2015	[413]
Golden Sine Algorithm	GSA.1	2017	[578]
Glowworm Swarm Optimization	GSO	2013	[236]
Galactic Swarm Optimization	GSO.2	2016	[414]
Goose Team Optimization	GTO	2008	[238]
Gorilla Troops Optimizer	GTO.1	2021	[239]
Group Teaching Optimization Algorithm	GTOA	2020	[489]
Grey Wolf Optimizer	GWO	2014	[240]
Hitchcock Birds-Inspired Algorithm	HBIA	2020	[241]
Hydrological Cycle Algorithm	HCA	2017	[415]
Hunger Games Search	HGS	2021	[243]
Harry's Hawk Optimization Algorithm	HHO	2019	[244]
Horned Lizard Optimization Algorithm	HLOA	2024	[246]
Heart Optimization	HO.1	2014	[580]
Hurricane Based Optimization Algorithm	HO.2	2014	[418]
Hybrid Rice Optimization	HRO	2016	[100]
Hunting Search	HuS	2010	[248]
Honeybee Social Foraging	HSF	2007	[249]
Humboldt Squid Optimization Algorithm	HSOA.1	2023	[252]
Heat Transfer Search Algorithm	HTS	2015	[459]
Human Urbanization Algorithm	HUA	2020	[493]
Ideology Algorithm	IA	2016	[466]
Imperialist Competitive Algorithm	ICA	2007	[467]
Ideological Sublations	IS	2017	[582]

Table 21: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by representative solutions (IV).

Influenced by representative solutions (IV)			
Algorithm Name	Acronym	Year	Reference
Interior Search Algorithm	ISA	2014	[583]
Jaguar Algorithm	JA	2015	[256]
Jellyfish Search	JS	2021	[257]
Kidney-Inspired Algorithm	KA.1	2017	[585]
Kinetic Gas Molecules Optimization	KGMO	2014	[462]
Krill Herd	KH	2012	[259]
Kho-Kho optimization Algorithm	KKOA	2020	[494]
Kookaburra Optimization Algorithm	KOA	2023	[260]
Krestrel Search Algorithm	KSA	2016	[261]
Killer Whale Algorithm	KWA	2017	[262]
Seven-Spot Ladybird Optimization	LBO	2013	[264]
Lyrebird Optimization Algorithm	LBO.1	2023	[265]
League Championship Algorithm	LCA.1	2014	[495]
Lotus Effect Algorithm	LEA	2023	[530]
Leaders and Followers Algorithm	LFA	2015	[497]
Literature Research Optimizer	LRO.1	2024	[588]
Lightning Search Algorithm	LSA	2015	[422]
Locust Swarms Optimization	LSO	2009	[269]
Leopard Seal Optimization	LSO.1	2023	[270]
Membrane Algorithms	MA	2005	[589]
Mayfly Optimization Algorithm	MA.1	2020	[272]
Mine Blast Algorithm	MBA	2013	[590]
Magnetotactic Bacteria Optimization Algorithm	MBO	2013	[273]
Mouth Breeding Fish Algorithm	MBF	2018	[276]
Modified Cuckoo Search	MCS	2009	[278]
Modified Cockroach Swarm Optimization	MCSO	2011	[279]
Moth Flame Optimization Algorithm	MFO	2015	[280]
Magnetic Optimization Algorithm	MFO.2	2008	[423]
Meerkats Inspired Algorithm	MIA	2018	[282]
Marine Predators Algorithm	MPA	2020	[285]
Mushroom Reproduction Optimization	MRO	2018	[105]
Monkey Search	MS	2007	[286]
Moth Search Algorithm	MS.2	2018	[287]
Mantis Search Algorithm	MSA	2023	[288]
Multi-Verse Optimizer	MVO	2016	[426]
Naked Moled Rat	NMR	2019	[290]
Nutcracker Optimization Algorithm	NOA	2023	[291]
Nizar Optimization Algorithm	NOA.1	2024	[592]
Nomadic People Optimizer	NPO	2019	[292]
Newton-Raphson-Based Optimizer	NRBO	2024	[427]
Orcas Intelligence Algorithm	OA	2020	[293]
OptBees	OB	2012	[294]
Optimal Foraging Algorithm	OFA	2017	[295]
Optics Inspired Optimization	OIO	2015	[428]
Owls Optimization Algorithm	OOA	2019	[296]
Osprey Optimization Algorithm	OOA.1	2023	[297]
Orca Predation Algorithm	OPA	2022	[298]

Table 22: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by representative solutions (V).

Influenced by representative solutions (V)			
Algorithm Name	Acronym	Year	Reference
Oriented Search Algorithm	OSA	2008	[499]
Prairie Dog Optimization Algorithm	PDO	2022	[302]
Paddy Field Algorithm	PFA	2009	[534]
Pigeon Inspired Optimization	PIO	2014	[303]
Population Migration Algorithm	PMA	2009	[304]
Political Optimizer	PO	2020	[500]
Puma Optimizer	PO.1	2024	[305]
Parliamentary Optimization Algorithm	POA	2008	[501]
Pelican Optimization Algorithm	POA.1	2022	[306]
Pufferfish Optimization Algorithm	POA.2	2024	[307]
Prey Predator Algorithm	PPA	2015	[308]
Plant Propagation Algorithm	PPA.1	2009	[533]
Poor and Rich Optimization Algorithm	PRO	2019	[502]
Particle Swarm Optimization	PSO	1995	[80]
Penguins Search Optimization Algorithm	PSOA	2013	[309]
Passing Vehicle Search	PVS	2016	[595]
Queuing Search Algorithm	QSA.1	2018	[503]
Regular Butterfly Optimization Algorithm	RBOA	2019	[310]
Artificial Raindrop Algorithm	RDA.1	2014	[596]
Red Fox Optimization Algorithm	RFO	2021	[312]
Root Growth Optimizer	RGO	2015	[535]
Roach Infestation Problem	RIO	2008	[315]
Radial Movement Optimization	RMO	2014	[435]
Ray Optimization	RO	2012	[436]
Red Piranha Optimization	RPO	2023	[318]
Red Panda Optimization Algorithm	RPO.1	2023	[319]
Runner Root Algorithm	RRA	2015	[537]
Raven Roosting Optimization Algorithm	RROA	2015	[320]
Red-tailed Hawk Algorithm	RTH	2023	[321]
Reptile Search Algorithm	RSA	2022	[322]
Rat Swarm Optimizer	RSO	2021	[323]
Root Tree Optimization Algorithm	RTOA	2016	[536]
Rain Water Algorithm	RWA	2017	[433]
Snow Ablation Optimizer	SAO	2023	[437]
Search and Rescue Algorithm	SAR	2019	[504]
Swarm Bipolar Algorithm	SBA	2024	[326]
Satin Bowerbird Optimizer	SBO	2017	[328]
Stem Cells Algorithm	SCA	2011	[108]
Sine Cosine Algorithm	SCA.2	2016	[329]
Social Cognitive Optimization	SCO	2002	[506]
Social Cognitive Optimization Algorithm	SCOA	2010	[507]
Sand Cat Swarm Optimization	SCSO	2023	[330]
Social Emotional Optimization Algorithm	SEA	2010	[508]
Stock Exchange Trading Optimization	SETO	2022	[509]
Synergistic Fibroblast Optimization	SFO	2017	[464]
Stochastic Focusing Search	SFS	2008	[510]
Stochastic Fractal Search	SFS.1	2015	[549]

Table 23: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by representative solutions (VI).

Influenced by representative solutions (VI)			
Algorithm Name	Acronym	Year	Reference
Space Gravitational Algorithm	SGA	2005	[438]
Soccer Game Optimization	SGO	2012	[511]
Spotted Hyena Optimizer	SHO	2017	[333]
Selfish Herds Optimizer	SHO.1	2017	[334]
Sea-horse Optimizer	SHO.2	2023	[335]
Swarm Inspired Projection Algorithm	SIP	2009	[336]
Soccer League Competition	SLC	2014	[468]
Slime Mould Algorithm	SMA	2008	[337]
Sperm Motility Algorithm	SMA.1	2017	[338]
Spider Monkey Optimization	SMO	2014	[339]
Starling Murmuration Optimizer	SMO.1	2022	[340]
States Matter Optimization Algorithm	SMS	2014	[440]
Spiral Dynamics Optimization	SO	2011	[441]
Student Psychology Optimization Algorithm	SPBO	2020	[512]
Spiral Optimization Algorithm	SPOA	2010	[442]
Self-Driven Particles	SPP	1995	[443]
Seeker Optimization Algorithm	SOA	2007	[341]
Seagull Optimization Algorithm	SOA.1	2019	[342]
Sandpiper Optimization Algorithm	SOA.2	2020	[343]
Sailfish Optimizer Algorithm	SOA.3	2019	[344]
Serval Optimization Algorithm	SOA.4	2022	[345]
Symbiosis Organisms Search	SOS	2014	[346]
Ship Rescue Optimization	SRO	2024	[553]
Siberian Tiger Optimization	STO	2022	[359]
Sooty Tern Optimization Algorithm	STOA	2019	[347]
Social Spider Algorithm	SSA	2015	[348]
Squirrel Search Algorithm	SSA.1	2019	[349]
Sparrow Search Algorithm	SSA.3	2020	[351]
Solar System Algorithm	SSA.4	2021	[444]
Shark Smell Optimization	SSO	2016	[353]
Swallow Swarm Optimization	SSO.1	2013	[354]
Social Spider Optimization	SSO.2	2013	[355]
Stadium Spectators Optimizer	SSO.3	2024	[513]
Sperm Swarm Optimization Algorithm	SSOA	2018	[356]
See-See Partridge Chicks Optimization	SSPCO	2015	[357]
Surface-Simplex Swarm Evolution Algorithm	SSSE	2017	[358]
Spider Wasp Optimizer	SWO.1	2023	[361]
Termite Alate Optimization Algorithm	TAOA	2023	[363]
T-Cell Immune Algorithm	TCIA	2023	[112]
Termite Colony Optimization	TCO	2010	[364]
Tasmanian Devil Optimization	TDO	2022	[365]
Tomtit Flock Optimization Algorithm	TFOA	2022	[366]
Team Game Algorithm	TGA	2018	[515]
The Great Salmon Run Algorithm	TGSR	2013	[367]
Teaching-Leaning Based Optimization Algorithm	TLBO	2011	[516]
Termite Life Cycle Optimizer	TLCO	2023	[368]
Tree Physiology Optimization	TPO	2018	[544]

Table 24: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by representative solutions (VII).

Influenced by representative solutions (VII)			
Algorithm Name	Acronym	Year	Reference
Thieves and Police Optimization Algorithm	TPOA	2021	[517]
Tyrannosaurus Optimization Algorithm	TROA	2023	[369]
Tree Seed Algorithm	TSA	2015	[545]
Tunicate Swarm Algorithm	TSA.1	2020	[370]
Tangent Search Algorithm	TSA.2	2022	[371]
Tiki-Taka Algorithm	TTA	2020	[514]
Tactical Unit Algorithm	TUA	2024	[518]
Tug Of War Optimization	TWO	2016	[519]
Unconscious Search	US	2012	[520]
Virus Colony Search	VCS	2016	[374]
Variable Mesh Optimization	VMO	2012	[113]
Volleyball Premier League Algorithm	VPL	2017	[521]
Vibrating Particle Systems Algorithm	VPO	2017	[446]
Vortex Search Algorithm	VS	2015	[447]
Wolf Colony Algorithm	WCA.1	2011	[378]
Water Cycle Algorithm	WCA.2	2012	[448]
Wind Driven Optimization	WDO	2010	[556]
Water Evaporation Optimization	WEO	2016	[449]
Whale Optimization Algorithm	WOA	2016	[380]
Walruses Optimization Algorithm	WaOA	2023	[381]
Wolf Pack Search	WPS	2007	[382]
Weightless Swarm Algorithm	WSA	2012	[383]
Wolf Search Algorithm	WSA.1	2012	[384]
White Shark Optimizer	WSO.1	2022	[386]
Water Wave Optimization Algorithm	WWA	2015	[453]
Yellow Saddle Goldfish	YSGA	2018	[387]
Zebra Optimization Algorithm	ZOA	2022	[388]
Zombie Survival Optimization	ZSO	2012	[389]

Table 25: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by subpopulations.

Influenced by subpopulations

Algorithm Name	Acronym	Year	Reference
Artificial Chemical Process	ACP	2005	[454]
Artificial Cooperative Search	ACS	2012	[558]
Artificial Physics Optimization	APO	2009	[397]
Bee Colony-Inspired Algorithm	BCIA	2009	[143]
Colliding Bodies Optimization	CBO	2014	[400]
Cuttlefish Algorithm	CFA	2013	[178]
Cuckoo Optimization Algorithm	COA	2011	[182]
Carnivorous Plant Algorithm	CPA	2021	[527]
Chicken Swarm Optimization	CSO.1	2014	[192]
COVID-19 Optimizer Algorithm	CVA	2020	[568]
Dice Game Optimizer	DGO	2019	[569]
Exchange Market Algorithm	EMA	2014	[572]
Greedy Politics Optimization Algorithm	GPO	2014	[487]
Gaining-sharing Knowledge	GSK	2023	[488]
Group Search Optimizer	GSO.1	2009	[237]
Horse Optimization Algorithm	HOA	2020	[247]
Hierarchical Swarm Model	HSM	2010	[250]
Ions Motion Optimization Algorithm	IMO	2015	[460]
Life Choice Based Optimizer	LCBO	2020	[496]
Lion Optimization Algorithm	LOA	2016	[267]
Monarch Butterfly Optimization	MBO.1	2017	[274]
Social Behavior Optimization Algorithm	SBO.1	2003	[505]
Sperm Whale Algorithm	SWA	2016	[360]
Thermal Exchange Optimization	TEO	2017	[465]
Turbulent Flow of Water-based Optimization	TFWO	2020	[445]
Wisdom of Artificial Crowds	WAC	2011	[522]
Worm Optimization	WO	2014	[379]

Table 26: Nature- and bio-inspired meta-heuristics within the *Differential Vector Movement* category, wherein the differential vector is influenced by neighborhoods.

Influenced by neighbourhoods

Algorithm Name	Acronym	Year	Reference
Bees Algorithm	BA	2006	[140]
Biomimicry Of Social Foraging Bacteria for Distributed Optimization	BFOA	2002	[148]
Bacterial Foraging Optimization	BFOA.1	2009	[62]
Gravitational Emulation Local Search	GELS	2009	[409]
Neuronal Communication Algorithm	NCA	2017	[591]
Physarum-inspired Competition Algorithm	PCA.1	2023	[301]

is that by combining good solutions, even better solutions can be eventually generated.

The combining method can be specific for the problem to be solved or instead, be conceived for a more general family of problems. In fact, combining methods are usually devised to be adaptable to many different solution representations. As mentioned before, the most popular algorithm in this category is GA [98]. However, many other bio-inspired algorithms exhibit a similar behavior when creating solutions, yet they are inspired by other phenomena, such as Cultural Optimization (CA, [479]) (in the Social Human Behavior category), LA [267] (in the Swarm Intelligence category), Particle Collision Algorithm (PCA, [429], in the chemistry-based category) or Light Ray Optimization (LRO, [421], in the physics-based category). Tables 27, 28, 29, and 30 show the algorithms that rely on combination when creating new solutions along their search.

4.2.2 Creation by Stigmergy

Another popular option of creating new solutions relies on stigmergy, namely, an indirect communication and coordination between the different solutions or agents used to create new solutions. This communication is usually done using an intermediate structure, with information obtained from the different solutions, used to generate new solutions oriented towards more promising areas of the search space. This is indeed the search mechanism used in the most representative algorithm of this category, ACO [598], which is inspired by the foraging mechanism of ant colonies. Each ant of the colony describes a trajectory over a graph representation of the search space of the problem at hand, and leaves a trace of pheromone along its way whose intensity depends, in part, on the fitness value corresponding to the solution encoded by the trajectory of the ant. In subsequent iterations, new solutions are generated, dimension by dimension, considering the pheromones trail left by preceding ants, enforcing the search around the most promising values for each dimension.

Table 31 lists the reviewed algorithms that employ stigmergy when creating new solutions. This is a reduced list when comparing with preceding categories, with the majority of the algorithms relying on Swarm Intelligence among insects (similarly to ACO). However, other algorithms inspired in physics have also a stigmertic behavior when producing new solutions, such as methods inspired by water flow dynamics [452] and the natural formation of rivers [434].

5 Taxonomies Analysis: Comparison and More Influential Algorithms

We now proceed by critically examining the reviewed literature as per the different taxonomies proposed in this overview. First, we are going to study the similarities between the results of the classifications following each taxonomy. Later, we identify the most influential algorithms over the rest, based on the behavior of the algorithms.

5.1 Comparison Between both Taxonomies

Comparing the two taxonomies to each other and the algorithms falling into each of their categories, it can be observed that there is not a strong relationship between them. Interestingly, this unveils that features characterizing one algorithm are loosely associated with its inspirational model. For instance, algorithms inspired by very different concepts such as the gravitational forces (GFA, [410]) or animal evolution (ABO, [123]) exhibit a significant similarity with PSO [80]. This statement is supported by the fact that, in the second taxonomy, each category is composed by algorithms that, as per the first taxonomy, are inspired by diverse phenomena. The contrary also holds in general: proposals with very similar natural inspiration fall in the same category of the first taxonomy (as expected), but their search procedures differ significantly from each other, thereby being classified in different categories of the second taxonomy. An illustrative example is the Delphi Echolocation algorithm (DE, [195]) and the Dolphin Partner Optimization [201]. Both are inspired by the same animal (dolphin) and its mechanism to detect fishes (echolocation), but they are very different algorithms: the former creates new solutions by combination, whereas the latter resembles closely the movement performed in the PSO solver, mainly guided by the best solution.

In this same line of reasoning, the largest subcategory of the second taxonomy (Differential Vector Movements guided by representative solutions) not only contains more than half of the reviewed algorithms (almost 60%), but it also comprises algorithms from all the different categories in the first taxonomy: Social Human Behavior (as Anarchic Society Optimization, ASO, [472]), microorganisms (Bacterial Colony Optimization, [145]), Physics/Chemistry category (correspondingly, Fireworks Algorithm Optimization, FAO, [575]), Breeding-based Evolution (as Variable Mesh Optimization, VMO [113]), or even Plants-Based (such as Flower Pollination Algorithm, FPA [529]). This inspirational diversity is not exclusive to this subcategory. Others, such as Solution Creation, also include algorithms relying on the heterogeneity of natural concepts.

Table 27: Nature- and bio-inspired meta-heuristics within the *Solution Creation - Combination* category (I).

Creation-Combination category (I)

Algorithm Name	Acronym	Year	Reference
Artificial Beehive Algorithm	ABA	2009	[121]
Andean Condor Algorithm	ACA	2019	[124]
Artificial Chemical Reaction Optimization Algorithm	ACROA	2011	[455]
Artificial Ecosystem Algorithm	AEA	2014	[82]
Artificial Flora Optimization Algorithm	AFO	2018	[524]
Artificial Infections Disease Optimization	AIDO	2016	[84]
Innovative Gunner Algorithm	AIG	2019	[559]
Anglerfish Algorithm	AOA	2019	[132]
Artificial Reaction Algorithm	ARA	2013	[456]
Asexual Reproduction Optimization	ARO	2010	[85]
American Zebra Optimization Algorithm	AZOA	2023	[138]
Bacterial-GA Foraging	BGAF	2007	[149]
Bumblebees	BB	2009	[141]
Biogeography Based Optimization	BBO	2008	[86]
Bee Colony Optimization	BCO	2005	[144]
BeeHive Algorithm	BHA	2004	[150]
Bees Life Algorithm	BLA	2018	[151]
Bird Mating Optimization	BMO	2014	[87]
Barnacles Mating Optimizer	BMO.1	2019	[155]
Bean Optimization Algorithm	BOA	2011	[88]
Bull Optimization Algorithm	BOA.1	2015	[159]
Bee System	BS	1997	[160]
Bar Systems	BS.2	2008	[563]
Backtracking Search Optimization	BSO.3	2012	[564]
Bees Swarm Optimization Algorithm	BSOA	2005	[166]
Bus Transportation Behavior	BTA	2019	[474]
BrunsVigia Flower Optimization Algorithm	BVOA	2018	[526]
Black Widow Optimization Algorithm	BWO	2020	[168]
Cultural Algorithms	CA	1999	[479]
Cultural Coyote Optimization Algorithm	CCOA	2019	[175]
Crystal Energy Optimization Algorithm	CEO	2016	[401]
Consultant Guide Search	CGS	2010	[179]
Coronavirus Mask Protection Algorithm	CMPA	2023	[89]
Coronavirus Disease Optimization Algorithm	COVIDOA	2022	[90]
Coral Reefs Optimization	CRO	2014	[91]
Chemical Reaction Optimization Algorithm	CRO.1	2010	[457]
Cuckoo Search	CS	2009	[188]
Clonal Selection Algorithm	CSA.1	2000	[567]
Dragonfly Swarm Algorithm	DA.1	2020	[194]
Dendritic Cells Algorithm	DCA	2005	[92]
Dolphin Echolocation	DE.1	2013	[195]
Discrete Mother Tree Optimization	DMTO	2020	[528]
Duelist Optimization Algorithm	DOA	2016	[480]
Dialectic Search	DS	2009	[570]
Election Algorithm	EA	2015	[481]
Ecogeography-Based Optimization	EBO	2014	[94]
Eco-Inspired Evolutionary Algorithm	EEA	2011	[95]
Electromagnetic Field Optimization	EFO	2016	[404]

Table 28: Nature- and bio-inspired meta-heuristics within the *Solution Creation - Combination* category (II).

Creation-Combination category (II)

Algorithm Name	Acronym	Year	Reference
Electric Fish Optimization	EFO.1	2020	[205]
Extremal Optimization	EO	2000	[573]
Equilibrium Optimizer	EO.1	2020	[574]
Earthworm Optimization Algorithm	EOA	2018	[96]
Electimize Optimization Algorithm	EOA.1	2011	[406]
Emperor Penguins Colony	EPC	2019	[209]
Evolution Strategies	ES	2002	[97]
Egyptian Vulture Optimization Algorithm	EV	2013	[214]
Frog Call Inspired Algorithm	FCA	2009	[217]
Forest Optimization Algorithm	FOA.1	2014	[523]
FOX-inspired Optimization Algorithm	FOX	2023	[223]
Genetic Algorithms	GA	1989	[98]
Golden Ball Algorithm	GBA	2014	[484]
Galaxy Based Search Algorithm	GBSA	2011	[393]
Gene Expression	GE	2001	[99]
Group Leaders Optimization Algorithm	GLOA	2011	[486]
Goat Search Algorithms	GSA.2	2022	[235]
Golf Sport Inspired Search	GSIS	2024	[579]
Honey-Bees Mating Optimization Algorithm	HBMO	2006	[242]
Hyper-parameter Dialectic Search	HDS	2020	[581]
Harmony Elements Algorithm	HEA	2009	[416]
Human Evolutionary Model	HEM	2007	[490]
Human-Inspired Algorithms	HIA	2009	[492]
Hysteresis for Optimization	HO	2002	[417]
Harmony Search	HS	2005	[394]
Hypercube Natural Aggregation Algorithm	HYNAA	2019	[253]
Japanese Tree Frogs Calling Algorithm	JTFCA	2012	[258]
Immune-Inspired Computational Intelligence	ICI	2008	[101]
Improved Genetic Immune Algorithm	IGIA	2017	[102]
Improved Raven Roosting Algorithm	IRRO	2018	[254]
Invasive Tumor Optimization Algorithm	ITGO	2015	[255]
Weed Colonization Optimization	IWO	2006	[103]
Keshtel Algorithm	KA	2014	[584]
Kaizen Programming	KP	2014	[586]
Lion Algorithm	LA	2012	[263]
Laying Chicken Algorithm	LCA	2017	[266]
Liver Cancer Algorithm	LCA.2	2023	[587]
Lion Pride Optimizer	LPO	2012	[268]
Light Ray Optimization	LRO	2010	[421]
Migrating Birds Optimization	MBO.2	2012	[275]
Migration-Crossover Algorithm	MCA	2024	[277]
Mosquito Flying Optimization	MFO.1	2016	[281]
Marriage In Honey Bees Optimization	MHBO	2001	[104]
Method of Musical Composition	MMC	2014	[424]
Mycorrhiza Optimization Algorithm	MOA	2023	[283]
Mox Optimization Algorithm	MOX	2011	[284]
Melody Search	MS.1	2011	[425]
Natural Aggregation Algorithm	NAA	2016	[289]

Table 29: Nature- and bio-inspired meta-heuristics within the *Solution Creation - Combination* category (III).

Creation-Combination category (III)

Algorithm Name	Acronym	Year	Reference
Natural Forest Regeneration Algorithm	NFR	2016	[531]
Old Bachelor Acceptance	OBA	1995	[498]
Photosynthetic Algorithm	PA	1999	[463]
Pity Beetle Algorithm	PBA	2018	[299]
Polar Bear Optimization Algorithm	PBOA	2017	[300]
Particle Collision Algorithm	PCA	2007	[429]
Plant Growth Optimization	PGO	2008	[532]
Plasma Generation Optimization	PGO.1	2020	[593]
Pearl Hunting Algorithm	PHA	2012	[594]
PopMusic Algorithm	PopMusic	2002	[430]
Queen-Bee Evolution	QBE	2003	[106]
Quantum Superposition Algorithm	QSA	2015	[431]
Red Deer Algorithm	RDA	2016	[311]
Reactive Dialectic Search	RDS	2017	[597]
Rain-Fall Optimization Algorithm	RFOA	2017	[432]
Rhino Herd Behavior	RHB	2018	[313]
Rock Hyraxes Swarm Optimization	RHSO	2021	[314]
Raccoon Optimization Algorithm	ROA	2018	[316]
Reincarnation Concept Optimization Algorithm	ROA.1	2010	[317]
Ringed Seal Search	RSS	2015	[324]
Shark Search Algorithm	SA	1998	[325]
Simulated Annealing	SA.1	1989	[79]
Scientifics Algorithms	SA.2	2014	[547]
SuperBug Algorithm	SuA	2012	[107]
Simulated Bee Colony	SBC	2009	[327]
Snap-Drift Cuckoo Search	SDCS	2016	[331]
Self-Defense Mechanism Of The Plants Algorithm	SDMA	2018	[539]
Social Engineering Optimization	SEO	2017	[548]
Sheep Flock Heredity Model	SFHM	2001	[109]
Shuffled Frog-Leaping Algorithm	SFLA	2006	[332]
Snow Flake Optimization Algorithm	SFO.1	2023	[550]
Saplings Growing Up Algorithm	SGA.1	2007	[538]
Search Group Algorithm	SGA.2	2015	[551]
Swine Influenza Models Based Optimization	SIMBO	2013	[110]
Sonar Inspired Optimization	SIO	2017	[439]
Seasons Optimization	SO.1	2022	[540]
Self-Organizing Migrating Algorithm	SOMA	2004	[111]
Simple Optimization	SOPT	2012	[552]
Strawberry Plant Algorithm	SPA	2014	[541]
Smart Root Search	SRS	2020	[542]
Salp Swarm Algorithm	SSA.2	2017	[350]
Sling-shot Spider Optimization	S^2SO	2023	[352]
Tree Growth Algorithm	TGA.1	2019	[543]
The Great Deluge Algorithm	TGD	1993	[555]
Small World Optimization	SWO	2006	[554]
Virulence Optimization Algorithm	VOA	2016	[114]
Virus Optimization Algorithm	VOA.1	2009	[375]
Viral Systems Optimization	VSO	2008	[376]

Table 30: Nature- and bio-inspired meta-heuristics within the *Solution Creation - Combination* category (IV).

Creation-Combination category (IV)

Algorithm Name	Acronym	Year	Reference
Wasp Colonies Algorithm	WCA	1991	[377]
Water Flow-Like Algorithms	WFA	2007	[450]
Water Flow Algorithm	WFA.1	2007	[451]
Wasp Swarm Optimization	WSO	2005	[385]
Ying-Yang Pair Optimization	YYOP	2016	[546]

Table 31: Nature- and bio-inspired meta-heuristics within the *Solution Creation - Stigmergy* category.

Solution Creation - Stigmergy

Algorithm Name	Acronym	Year	Reference
Ant Colony Optimization	ACO	1996	[115]
Bee System	BS.1	2002	[161]
Hammerhead Shark Optimization Algorithm	HSOA	2019	[251]
Intelligence Water Drops Algorithm	IWD	2009	[420]
River Formation Dynamics	RFD	2007	[434]
Termite Hill Algorithm	TA	2012	[362]
Virtual Ants Algorithm	VAA	2006	[372]
Virtual Bees Algorithm	VBA	2005	[373]
Water-Flow Algorithm Optimization	WFO	2011	[452]

Considering the previous examples, it is clear that the real behavior of the algorithm is much more informative than its natural or biological inspiration. Even more, we have observed that in our first proposed taxonomy, built upon the review of 518 proposals, the huge diversity of inspirational sources does not correspond with the lower number of algorithmic behaviors on which our second taxonomy is based. This observation is in accordance with previous works in the literature, which have put to question whether the novelty in the natural inspiration of the algorithm actually yields different algorithms that could produce competitive results [599, 600].

We further elaborate on the above statement: our literature analysis revealed that the majority of proposals (more than a half, 60%) generate new solutions based on differential vector forces over existing ones, as in the classical PSO or DE. A complementary analysis can be done by departing from this observation towards discriminating which of the classical algorithms (PSO, DE, GA, ACO, ABC or SA) can be declared to be most similar to modern approaches. The results of this analysis are conclusive: 23% of all reviewed algorithms (122 out of 518) were found to be so influenced by classical algorithms that, without their biological inspiration, they could be regarded as incremental variants. The other 396 solvers (about 77%) have enough differences to be considered a new proposal by themselves, instead of another version of existing classical algorithms. But, we must emphasize that in these new algorithms there exists a lack of originality or justification in a significant percentage of cases. We must emphasize that in these new algorithms there exists a lack of justification due to the lack of comparison with the state of the art and the lack of real interest in achieving reasonable levels of quality from the perspective of the optimization of well-known problems in recent competitions.

Table 32: Percentages of similar algorithms in the reviewed literature.

Classical algorithm	Number of papers with similar algorithms	Percentage over the total
PSO	57	11.00%
DE	24	4.63%
GA	24	4.63%
ACO	7	1.35%
ABC	7	1.35%
SA	3	0.59%
Total	122	23.55%

5.2 Identification of the Most Influential Algorithms

In order to know which are the most influential reference algorithms used to design other bio-inspired algorithms, we have grouped together reviewed proposals that could be considered to be versions of the same classical algorithm. Figure 5.2 shows the classification of each algorithm based on its behavior, and the number of proposals in each classification are summarized in Table 32.

Very insightful conclusions can be drawn from this grouping. To begin with, in Table 32 the most influential algorithm was identified to be PSO, appearing in 11% of the reviewed literature (which corresponds to almost 47% of the proposals that were clearly based on a previous algorithm). This bio-inspired solver is one of the most prominent and historically acknowledged algorithms in the Swarm Intelligence category and is the reference of many bio-inspired algorithms contributed since its inception. The simplicity of this algorithm and its ability to reach an optimum quickly – as has been comparatively assessed in many application scenarios, see e.g. [72, 73] – have inspired many authors to create new metaheuristics characterized by similar solution movement dynamics to those defined by PSO. Thus, many algorithms whose authors claim to simulate the behavior of a biological system eventually perform their search process through movements strongly influenced by PSO (in some cases, without any significant difference).

The second and third most influential algorithms are GA, a very classic algorithm, and DE, a well-known algorithm whose natural inspiration resides only in the evolution of a population. Both have been used by around 5% of all reviewed nature-inspired algorithms, and they are the most representative approach in the *Evolutionary Algorithms* category. The search mechanism of GA is solution creation by combination, and the search mechanism of DE is to create new solutions with a linear combination of existing ones in the population, which is used by 5% of all reviewed proposals, maybe by its superior

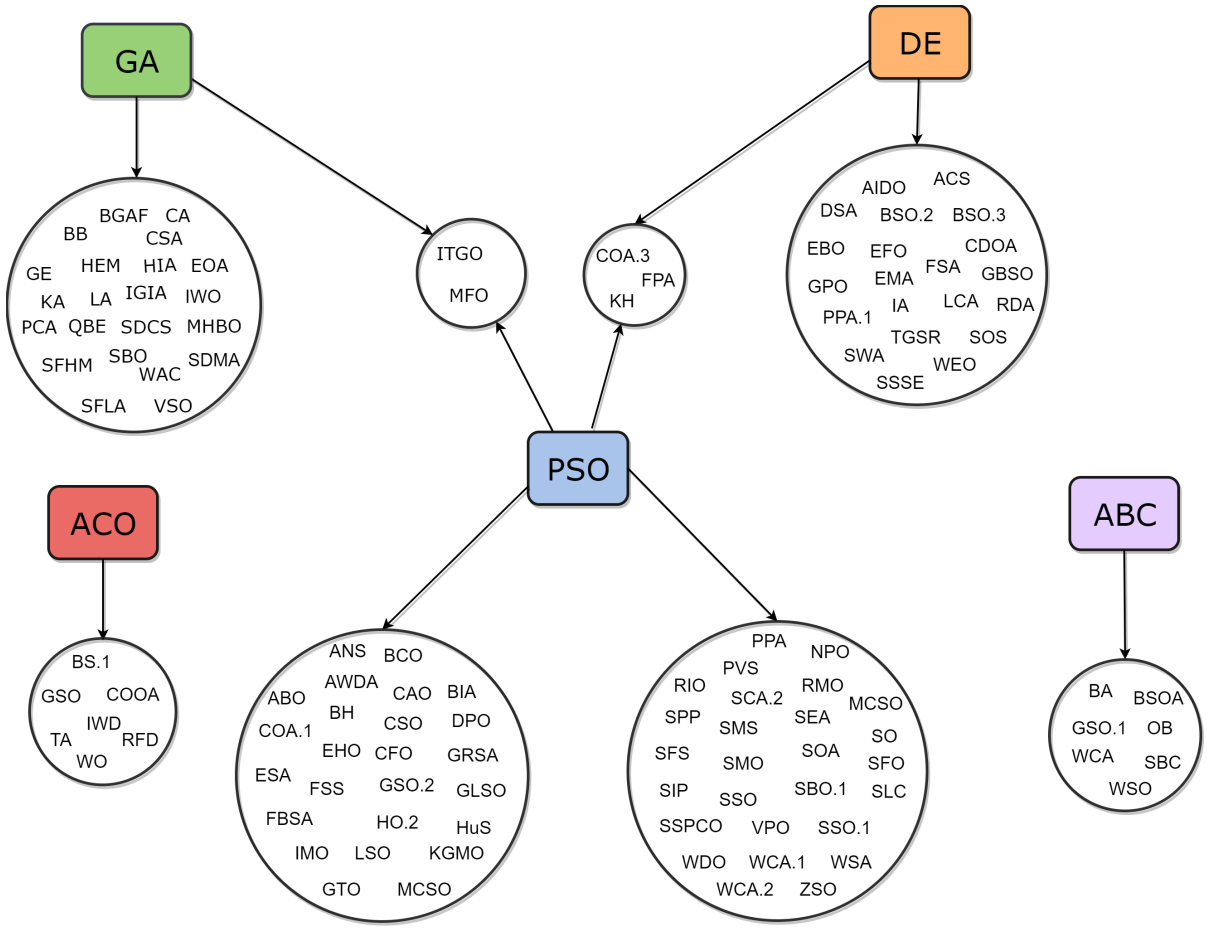


Figure 6: Classification of proposals by its original algorithm.

performance reported for many optimization problems [44].

When inspecting the influential approaches from a higher perspective, two are the categories whose algorithms have been more frequently used to create new nature-based algorithms. The first one is *Swarm Intelligence*: about 14% of all studied nature-inspired algorithms are variations of SI algorithms (PSO, ACO, and ABC). The second one is shared between *Evolutionary Algorithms* and GA, whose represented algorithms are both used in almost 5% of the reviewed cases. It is noteworthy to highlight that it appears that the influence of more classic algorithms like GA and SA is declining when compared to other algorithms, such as DE and PSO.

In summary, although in the last years many nature-inspired algorithms have been proposed by the community and their number grows steadily every year, more than half of the proposals reviewed in our work are incremental, minor versions of only three very classical algorithms (PSO, DE, and GA). We, therefore, conclude that a huge number of natural and biological sources of inspiration used so far to justify the design of new optimization solvers have not led to significantly disruptive algorithmic behaviors. This closing note will be at the heart of our critical analysis exposed in the next section.

6 Learned Lessons and Recommendations from the Analysis of the Evolution of Bio-Inspired Optimization

After reviewing the algorithms and both taxonomies, we have identified several key learned lessons which serve as recommendations to deal with in forthcoming years for that is working on nature- and bio-inspired optimization. The learned lessons gained from the taxonomies and research outlined in [1] form the foundation of this section. In the subsequent sections, we will further expand upon them to provide a more extended analysis. We next outline them in no particular order:

1. **The behavior is more relevant than the natural inspiration:** As was exposed in Section 5, the current literature is flooded with a huge number of nature- and bio-inspired algorithms. However, as has been spotted by our proposed taxonomies, several algorithms belonging to categories with different sources of inspiration results are very similar in terms of behavior. This disparity is a controversial topic in recent years [3, 599]. Therefore, we call for more research efforts around the design of optimization algorithms that focus on their behavior and properties (e.g., good performance, simplicity, ability to run it in parallel or their suitability to a specific type of problems) rather than on new inspiration sources.

2. **Nature-based terminology can make it more difficult to understand the proposal:** A great deal of papers presenting new bio-inspired solvers are difficult to understand and replicate due to the extended usage of vocabulary related to the natural source of inspiration. It is logical to use the semantic of the biological or natural domain, but to an extent. It would be desirable that the description of the algorithm could be defined in an inspiration-agnostic fashion, resorting to mathematical terms to describe each component, agent and/or phase of the optimization process (e.g. optimum/a, individuals, or solutions). Excessive usage of the domain terminology (without explicitly indicating the correspondences) could make it difficult to follow the details of the algorithm for researchers not acquainted with such a terminology. To overcome this issue, the correspondence between the domain terminology and the optimization terminology should be explicitly indicated.

3. **Good comparisons are crucial for new proposals:** The lack of fair comparisons is another important drawback of many proposals published to date. When new algorithms are proposed, unfortunately, many of them are only compared to very basic and classical algorithms (such as GA or PSO). These algorithms have been widely surpassed by more advanced versions over the years which, so obtaining better performance than naive version of classical algorithms is relatively easy to achieve, and it does not imply a competitive performance [600]. In some cases, the proposed algorithm is compared to similar algorithms but not with competitive algorithms outside that semantic niche [600, 601]. This methodological practice must be regarded as a very serious barrier for their application to real-world problems. We encourage researchers to increase the algorithms used in their experimental section, including more competitive or state-of-the-art algorithms: until they are proven to be competitive in respect to the state of the art, new nature- and bio-inspired solvers will not be used in practice either will attract enough attention of the research community.

4. **Many proposals have a very limited influence:** By examining in depth the historical trajectory followed by each reviewed algorithm, an intriguing trend is revealed: a fraction of the proposals have a very limited influence in new papers after the original publication. For them, there are almost no new papers with improved versions, or applying it to new problems.

Fortunately, other algorithms have a stronger influence. In view of this dichotomy, the researchers should evaluate their proposals for diverse problems, including widely acknowledged benchmark functions and real-world practical problems, to grasp the interest of the community in considering their proposed algorithms for tackling other applications.

5. **The interest of making source code available:** Related to the previous one, it is very interesting, in order to gain more visibility, to make the source code of the proposed algorithm available for the community. It is true that the paper presenting the new algorithm should be detailed enough to allow for a clean implementation of the proposal from the provided specification. However, it is widely acknowledged that, in many occasions, there are important details that even though they have a strong influence on the results, are not remarked in the description [602, 603]. A publicly available reference implementation could not only improve its visibility, but could also offer other researchers the chance to undertake more thorough performance comparisons. In addition, there are a huge number of software frameworks for Evolutionary Computation and Swarm Intelligence programmed in different languages (such as C++, Java, Matlab, or Python), some of them very popular in the current research landscape. To cite a few: Evolutionary Computation Framework (ECF)[1] and ParadisEO [604] in C++; jMetal [605] and MOEA [2] in Java; NiaPy[606], jMetalPy [607] and PyGMO[3] in Python; or PlatEMO [608] in Matlab, among others. Each of them implements the most popular algorithms (GA, DE, PSO, ABC, ...). A reference implementation could also favor the inclusion of the proposal in frameworks as the ones exemplified previously. Otherwise, different implementations of the allegedly same algorithm could produce diverging results from the original proposal (in part due to the ambiguity of the description).

6. **The role of bio-inspired algorithms in competitions:** Finally, we also stress on the fact that metaheuristic algorithms that have scored best in many competitions are far from being biologically inspired, although some of them retain their nature-inspired roots (mostly, DE) [44]. This fact was expected for the lack of good methodological practices when comparing nature- and bio-inspired algorithms, which was pointed out previously in our analysis. This issue has not encouraged participants in competitions to embrace them as reference algorithms to design better solvers. The rising trend of the community to generate an ever-growing number of bio-inspired proposals can be counterproductive and deviate efforts towards the development of a reduced number of proposals but with a better performance.

7 A Short Reflection on The *Good*, the *Bad* and the *Ugly*

This section corresponds to the integration and extension of Section 3 of the article published in [2] within this report. In Section 7.1, we extend the original analysis on the importance of applications, stressing on the numerous applications that leverage results from this research area (the *good*). In section 7.2, we have also extended the original content to more studies based on the recent problems of the area, namely, the lack of algorithmic innovation in algorithms inspired by novel metaphors and good comparisons between algorithms (the *bad*). Section 7.3 remains as in the original work, underscoring the poor practices experienced by the area in recent times (the *ugly*).

7.1 The *Good*: A Present and Future Plenty of Exciting Applications

An undeniable fact is that nature- and bio-inspired optimization algorithms have been applied to a great variety of optimization problems emerging in different disciplines. We distinguish among three different horizons of applications, without being exhaustive, nor entering into the recent horizons of general-purpose AI that we will mention in the conclusions. They are outlined shortly as follows:

- **Real-world engineering applications**: We can find many examples regarding the usage of bio-inspired techniques to solve real-world engineering processes [609]. Furthermore, structural design and civil engineering have also largely embraced the benefits of nature and bio-inspired solvers to assorted problems, including the multi-criteria design of structures [610], logistics and supply chain management [611], to cite a few. The application of Evolutionary Algorithms (EAs) has reached many areas, including works from these human competitions for the design of breakwaters [612], the evolution of antennas for Space Mission of the NASA [613], and also the discovery of new formulas in the field of physics [614], among many other important applications.

[1] http://ecf.zemris.fer.hr/
[2] http://moeaframework.org/
[3] http://esa.github.io/pygmo/index.html

- **Academic competitions**: From the research perspective, several worldwide competitions have developed over the years to test new proposals in an unbiased and replicable way. In such competitions, DE has created a great impact as the core meta-heuristic algorithm of winning competitors in the global optimization competitions held in renowned conferences (GECCO and CEC) over the last decade [44]. The family of EAs has attracted the interest of researchers by participating in genetic and evolutionary competitions with prizes [4] (Annual "Humies" Awards For Human-Competitive Results), and others such as GECCO and CEC previously annotated [600].

- **Going deeper into the creation of Machine Learning (ML) and Deep Learning (DL) models**: Although most algorithms have been developed in recent years, the impact of EAs, a classical family of algorithms, has risen in the last few years. Their use in ML has been widely studied both for the design of models [615] and also as a support for the optimization of those models [616]. These algorithms have gained momentum under the evidence reported around their usage to evolve and improve other AI techniques: most notably, the optimization of the structure and training parameters of deep neural networks [8], or the creation of new data-based models from scratch (i.e. by evolving very essential data processing primitives) that has been presented in the groundbreaking work by Google [617]. With this ongoing development, the research trend of Neural Architecture Search has emerged as another important area full of EAs applications [618], which mainly focuses on the construction of the DL model via the evolution of block of layers [619, 14, 620]. Recently, we have witnessed the use of EAs to model more AI models, as in the case of POET [621] where more environments are generated to learn from the diversity created, with the merging of EAs with Large Language Model (LLM) [622], and with other areas such as Automated Machine Learning [623], Reinforcement Learning and robotics [624], and Multi-task Learning [625]. In recent years, an interesting synergy between bio-inspired optimization and modern ML systems has been observed in the literature, in particular General-Purpose Artificial Intelligence Systems (GPAIS), as we will highlight later in the report.

7.2 The *Bad*: Novel Metaphors Not Leading to Innovative Solvers

As previously mentioned, an ever-growing amount of new bio-inspired optimization techniques has been proposed in recent decades (see Figure 1). This overwhelming number of alternatives could make it difficult to choose an appropriate option for a given optimization problem. The vast number of proposals not only casts doubt on the convenience of choosing one or another algorithm but has also produced solvers that, even if relying on different metaphors, are mathematically too similar to already existing optimization algorithms. In other words, despite the diversity of methods considering their natural inspiration, such diversity does not hold as far as mathematical differences are concerned, as exposed by recent studies [13]. As we have mentioned in the introduction, this metaphor-driven research trend has been already denounced in several contributions [9, 10], which have unleashed hot debates around specific meta-heuristic schemes that remain unresolved to date [11, 12], and with a growing problem when important challenges are not addressed and if more and more biological inspirations are maintained as we can observe in 2024 with more than 500 proposals.

Particular reasons aside, some algorithms are not created to solve problems and provide a practical advantage, but mainly to be published and gain notoriety without any consideration for their lack of algorithmic novelty and innovation. Examples of this controversy can be found in [14], as authors state this problem even in the title of the work. In the previous work, authors "provide compelling evidence that the grey wolf, the firefly, and the bat algorithms are not novel, but a reiteration of ideas introduced first for particle swarm optimization and reintroduced years later using new natural metaphors". Then, they rewrite these highly cited papers in terms of PSO, and conclude that "they create confusion because they hide their strong similarities with existing PSO algorithms ... these three algorithms are unnecessary since they do not add anything new to the tools that can be used to tackle optimization problems".

More and more works lack variety in the field, as it was discussed in [15] ("Nature inspired optimization algorithms or simply variations of metaheuristics?"), authors discussed several matters listed as follows:

- **Does the physical analogue exist?**: The inspiration of several bio-inspired algorithms does not strictly follow the rules of a phenomenon. An example is Cat Swarm Optimization, in which cats form a swarm, but in real life, they do not seem to cooperate in any way. Authors show more examples (Coyote Optimization Algorithm, Dolphin Swarm Optimization Algorithm, among others), and claim that " a significant number of these algorithms are very similar to other already existing ones".

[4] https://human-competitive.org/

- **Similar inspiration or duplicate methods?**: Authors analyze several classes of bio-inspired algorithms such as those based on gravitational forces, water phenomena, bees, penguins, wolves, and bacteria, and conclude that not all the different variations are real contributions.

- **Do authors propose multiple techniques based on the same idea?**: Authors discuss the fact that " several cases can be found where the same authors propose multiple algorithms, which are based on the same nature-inspired idea." They show various examples in which a research group has almost a dozen "novel" algorithms, with the same researchers at the front. Also, a relevant group of algorithms that are based and attraction and repulsion is full of works under the same researcher's name.

- **When should a new nature-inspired algorithm be introduced?**: The authors analyze the cases in which it is necessary to create novel algorithms. In their words, "They could be used as global optimizers, while a heuristic algorithm could be added for acting as local search technique for the solutions provided by the nature-inspired method." They also annotate the ability of these algorithms as optimizers for Artificial Neural Networks and Support Vector Machines.

Due to "useless metaphors", "lack of novelty" and "poor experimental validation and comparison", in [16] authors took the decision in this letter to "call upon all editors-in-chief in the field to adapt their editorial policies" to reject the publication of *novel* metaphor-based metaheuristics. More than 80 important researchers in the area signed this letter, and accept the publication of novel bio-inspired algorithms if and only if (1) present their method using the normal, standard optimization terminology; (2) show that the new method brings useful and novel concepts to the field; (3) motivate the use of the metaphor on a sound, scientific basis; and (4) present a fair comparison with other state-of-the-art methods using state-of-the-art practices for benchmarking algorithms.

In the following, we shortly describe the critical analysis that has recently been published in several articles that address this problem "not leading to innovative solvers":

- In [17], the authors argue that metaheuristics should be simplified by eliminating the unneeded elements as in the case of two winners of the CEC2016 competition, L-SHADE-EpSin and UMOEA-II. The authors conclude that these algorithms "contain operators that structurally bias their search by favouring sampling from some parts of the decision space" and "other metaheuristics should be simplified as they contain unneeded or even harmful operators". By doing so, metaheuristics will be easier to understand for other researchers. The authors simplify both algorithms by removing operators that are the main cause of structural bias and the experiments when testing against other metaheuristics reveal " that simplification of some metaheuristics may not only make them more transparent and easier to use, but also improve their performance."

- In [18, 19], the authors analyze the algorithm called Intelligent Water Drops, providing several proofs that " all main algorithmic components of Intelligent Water Drops are simplifications or special cases of ant colony optimization (ACO)". They also examine the natural metaphor of "water drops flowing in rivers removing the soil from the riverbed", which is the source of inspiration for this algorithm. Authors conclude that it "is unnecessary, misleading and based on unconvincing assumptions of river dynamics and soil erosion that lack a real scientific rationale".

- In [20], authors present an analysis of the Cuckoo Search, one of the most well-known algorithms in the literature. Their review of this algorithm based on its usefulness, novelty and sound motivation allow them to "conclude that neither the metaphor nor the algorithm can be considered as part of the set of useful techniques in stochastic optimization". The Cuckoo Search is just an evolutionary strategy with some parts of DE, algorithms from the last century.

- In [21], authors perform a comparison between seven bio-inspired algorithms with various benchmarks and discovered that "these (algorithms) contain a centre-bias operator that lets them find optima in the centre of the benchmark set with ease". The conclusion is that making more "comparison with other methods (that do not have a centre-bias) is meaningless". This problem is similar to the appearance of harmful operators, which has already been discussed in [17]. Authors carry out experimentation with these algorithms against DE and PSO on shifted problems and encounter that "the worst one performed barely better than a random search", which is a very serious problem.

- In [22], authors discuss the possible causes of the exponential growth of nature-inspired algorithms and the negative consequences for the field. One cause is the pressure to "publish or perish," and authors argue that the "publishing metaphor-based method is perceived as a low-effort, low-risk process with high potential rewards" because there are authors that have built professional careers out of creating not one but often multiple metaphor-based methods. The other cause

reflected by the authors is "the lack of a well-established statistical tradition in the field compounds the problem, leading to generally poor practices by authors and, in many cases, an inability of reviewers to pick up on the main methodological problems of some papers".

- In [23], authors aim to present some nature-inspired methods that contribute to achieving lifelike features of computing systems such as open-ended evolution, intelligence, emergence, resilience, and social awareness. In this work, authors select the algorithms of Big Bang–Big Crunch, Mine Blast Algorithm, Lightning Search Algorithm, Water Wave Optimization, Gravitational Search Algorithm, Cat Swarm Optimization, Chicken Swarm Optimization, and Roach Infestation Optimization to "investigate if the mechanisms being part of the algorithms produce qualities found in evolutionary, physical, or chemical analogues." The conclusion is that most nature-inspired algorithms "do not contribute to achieving lifelike features" and that "the recent algorithms do not remain accurate to the behavior or the phenomenon on which they are based."
- In [24], the authors claim that grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms are not novel algorithms, and their inspiration has been in the literature for years. To assert this, the authors present a rigorous, component-based analysis of each algorithm that reveals evidences about them: these algorithms are variants of PSO and evolutionary strategies.
- In [25], authors discuss the problem of centre-bias. 47 of 90 algorithms that were compared presented a centre-bias. Also, the authors conclude that Harmony Search (HS), Cuckoo Search Algorithm, Firefly Algorithm, Moth Flame Optimization, Ant Lion Optimizer (ALO) should not be used, due to similarities to other algorithms.

7.3 The *Ugly*: Poor Methodological Practices (Questionable Reproducibility and Comparability)

An alarming issue that prevails in the area besides the number of metaphor-based proposals is the lack of a fair experimental study to prove their competitiveness when compared to existing solvers. In many research contributions, the newly introduced bio-inspired optimization algorithms are not compared to relevant techniques, but only to classical solvers already surpassed by more recent approaches. Therefore, improving their performance in a benchmark is not a reliable proof of performance competitiveness, but rather a convenient choice of comparison counterparts. Moreover, the experimental design is often not right: for example, the optima of the tested functions is often at the center of the domain search, which favors solvers that focus their search over this region of the solution space. In addition, the statistical significance of the performance gaps reported among algorithms is also frequently overlooked, despite the variability of the results imprinted by the stochastic nature of these algorithms. In this regard, in [25] the same controversy as shown in [21] is followed: several algorithms contain a centre-bias operator that makes them more suitable for certain fitness functions. As a result of such bias, these algorithms achieve better results than other algorithms in the appearance of this condition. Thus, they should not be recommended for real-world problems, because the experiments that showed their good performance are biased.

Another important concern in the area is the questionable reproducibility of published studies: the only proof that a proposal is competitive is done experimentally, so it is of utmost importance that results can be reproduced, checked, and verified by third parties, ideally by a different team to that proposing the new algorithm. Unfortunately, in the majority of cases, this is not possible because the implementation of the algorithms is not available, or because important information for the replicability of the experiments is missing or not reported whatsoever [626].

More and more researchers are advocating that a novel metaphor is not enough for a new bio-inspired algorithm to be considered a step beyond the state of the art. Instead, several factors should be proven with empirical evidence, such as superior performance to the state of the art, innovation in the design of its mathematical components and operators, or non-functional benefits that make them more appropriate for real-world optimization problems when compared to other alternatives, e.g. less computational complexity, smaller memory footprint, or faster convergence properties [5].

We strongly urge interested readers to embrace the methodological practices recommended in [4], considering proposals that have been tested against modern techniques, using standard benchmarks, and with adequate statistical testing to shed light on the relevance of performance gaps. Unfortunately, many recent proposals do not follow these guidelines, remaining as evidence of the ugly side that still prevails in this research area.

8 Three Propositional Discussions about Nature- and Bio-Inspired Optimization

As we have mentioned in the introduction, we revisit a triple study of evolutionary and bio-inspired algorithms from a triple perspective, where we stand and what's next from a perspective published in 2020, but still valid in terms of the

need to address important problems and challenges in optimization for EAs and population-based optimization models, a prescription of methodological guidelines for comparing bio-inspired optimization algorithms, and a tutorial on the design, experimentation, and application of metaheuristic algorithms to real-world optimization problems.

8.1 Bio-inspired computation: Where we stand and what's next

We should pause and reflect on which research directions should be pursued in the future in regard to bio-inspired optimization and related areas, as there are other remarkable fields to be noted as direct applications for bio-inspired optimization. In [3], the authors show a full discussion of the status of the field from both descriptive (*where we stand*) and prescriptive (*what's next*) points of view. Here, we describe the areas in which bio-inspired optimization algorithms are used, and research niches related to them, as shown in Figure 7. The areas and their main aspects that can be studied as promising research lines are:

- Theoretical studies: By the hand of the fitness landscape for a better understanding of how a search algorithm can perform on a family of problem instances, of multidisciplinary theories to study the role of diversity and the balance of local search and global search required to undertake a certain problem efficiently, and of convergence, studies to identify the conditions for the convergence of the algorithm, its speed, fitness stability, and other characteristics.
- Dynamic and stochastic optimization: These areas need reliable modeling of real optimization scenarios where the characteristics of several of these problems hold (diversity control), and the development of change detection mechanisms relying on characteristics of the optimization algorithm (change detection).
- Multi/Many-objective optimization: These areas need new ideas regarding the design of multiobjective solvers because they usually use the main multi-objective solver (radically new approaches) or even the combination of different solvers to create new ones (hybridization of techniques). Another problem to be addressed is the scalability, as solvers do not scale properly with many objectives.
- Multimodal optimization: The incorporation of new bio-inspired multimodal solvers and the hybridization of new bio-inspired techniques with traditional strategies can contribute to the progress of this area.
- Topologies: A promising research direction is to jointly consider topologies and ensemble strategies to leverage the superior explorative/exploitative powers of ensembles and also topologies for population-based metaheuristics to achieve better solutions than other solvers.
- Surrogate model-assisted optimization: This area has promising research lines of investigation with highly dimensional search spaces and DL models, where there is a need to alleviate high computational efforts, with evaluation times that range from hours to days per experiment.
- Distributed EAs: These algorithms are needed in large-scale data mining to deal with expensive objective functions, which are common in real-world applications comprising multiple criteria, and also in large-scale multi-objective optimization for the development of asynchronous parallel multi-objective solvers.
- Ensemble methods and hyper-heuristics: Both areas have promising challenges in large-scale optimization to address problems such as the encoding strategy, the exploration capabilities of the algorithms, and the computational complexity of the proper ensembles. In real applications, these areas need to address the challenge of the appropriate selection of their low-level composing pieces.
- Memetic algorithms: Although these algorithms have shown great results, researchers need to investigate their hybridization with other bio-inspired optimization algorithms for the design of new algorithms, and also the derivation of self-adaptive mechanisms to tune the balance between exploration and exploitation
- Large-scale global optimization: For this area, it will be interesting to develop new techniques that automatically infer relationships among variables (grouping variables) that could be optimized in isolation with the minimum loss of efficiency and to study new approaches of memetic computing with this area, due to the great results of DE and large-scale global optimization.

- Parameter tuning: The assignation of proper values to the parameters of bio-inspired algorithms is crucial for obtaining the best possible results for a given problem, so the development of parametric sensitivity analysis, or robustness studies, can be very useful to identify the relevant parameters to tune. Also, when an algorithm is compared to another, it will be necessary to perform a similar parameter-tuning process to make fair comparisons.
- Parameter adaptation: Further research about self-adaptation mechanisms for very sensible parameters is necessary, as it reduces the parameters to tune, but can also yield great improvements.
- Benchmarks and comparison methodologies: The development of a novel bio-inspired solver includes the comparison to other techniques with several fitness functions. To encourage better comparison methodologies, the most promising avenues are the use of existing benchmarks and also the creation of new ones based on real-world problems. Moreover, better comparison methodologies, including more attention to scalability and new statistical testing approaches such as the use of Bayesian tests, are needed. We delve deeper into this in Subsection 8.2.

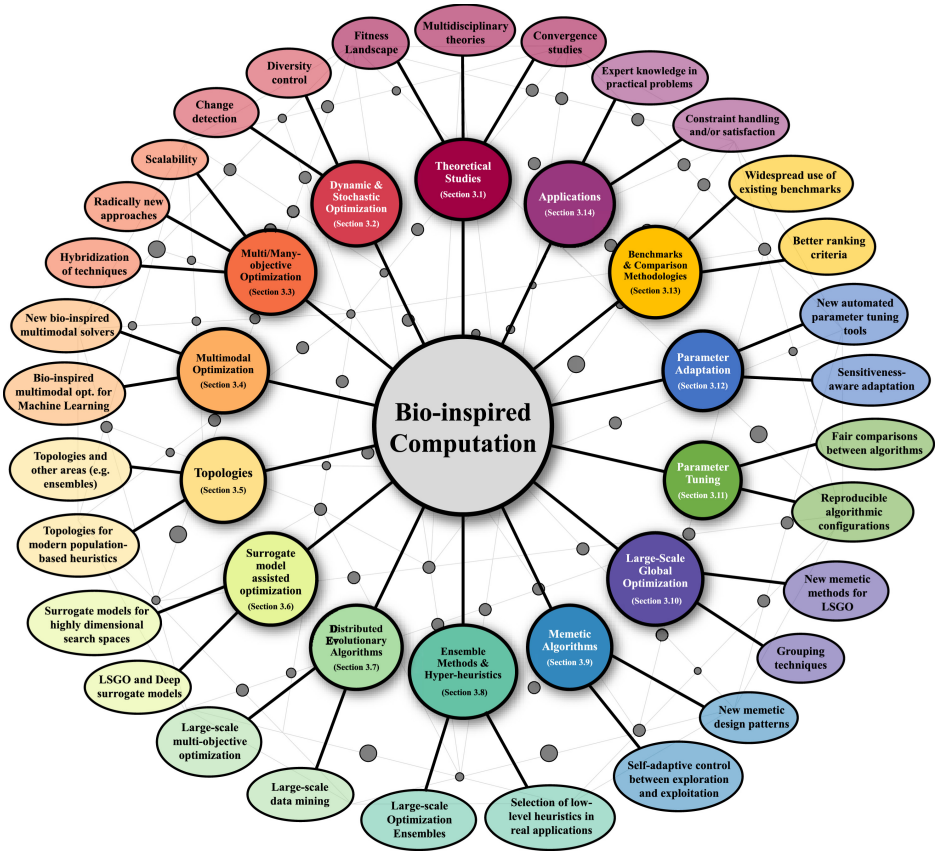


Figure 7: Bio-inspired optimization areas and promising research lines. Image taken from [3].

8.2 Separating the Wheat from the Chaff: Fair and Right Comparisons

One of the problems identified in this manuscript is the abundance of proposals with limited impact. A key aspect for these algorithms to show their strengths is the development of comparative best practices against more competitive algorithms and the state of the art.

To clarify and provide guidelines for a fair and effective comparison between bio-inspired proposals, an extended discussion of the various guidelines to be followed is presented in [4] and here it is summarized as follows:

1. **Benchmarks**: The choice of benchmarking in algorithm evaluation can vary between real-world scenarios and comparisons against existing algorithms. Selecting the right benchmark is crucial, as study conclusions heavily rely on the test environment. However, chosen benchmarks often exhibit biases that can unfairly advantage certain algorithms. Consequently, it is essential to analyze results considering the diverse characteristics of the test problems within the chosen benchmark to ensure fairness in subsequent comparisons.

2. **Validation of the results**: Today, simply presenting raw results in extensive tables falls short. Validating results statistically is imperative, complementing tables with proper statistical analyses. It is crucial to not just employ statistical tests, but to ensure they are appropriate for the data at hand. Often, parametric tests are used without verifying if the underlying assumptions are met by the results. Moreover, the use of visualization techniques in comparative analysis is crucial, as these methods condense vast amounts of data into easily comprehensible representations, also aiding quick interpretation for readers.

3. **Components analysis and parameter tuning of the proposal**: The hypotheses of the proposal should be explicitly outlined at the paper's outset and revisited upon validation of results. Furthermore, authors ought to undertake a comprehensive analysis of results, addressing key aspects such as search phase identification (balancing exploration and exploitation), component analysis (individually assessing each method component and its complexity), algorithm parameter tuning, and statistical comparison with state-of-the-art algorithms. This thorough examination ensures a robust evaluation of the proposed method and its performance relative to existing approaches.

4. **Why is my algorithm useful?**: Prospective contributors must articulate why their proposed algorithm merits attention within the community. Several reasons are proposed to show why a new proposal constitutes an advancement in knowledge, such as its competitiveness against state-of-the-art methods or methodological contributions that foster additional research. This clarity aids in understanding the significance of the proposed algorithm and its potential impact on the field.

These four guidelines form the basis of their work and are discussed in more detail inside the paper published in [4]. Finally, these guidelines are further detailed in Figure 8.

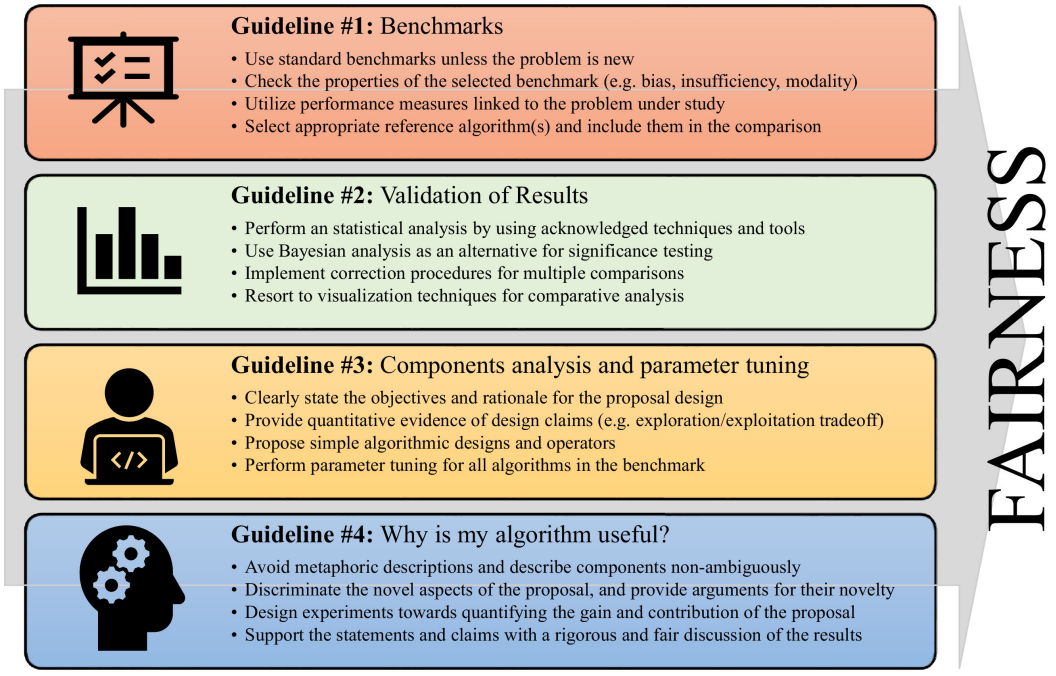


Figure 8: Summary of the guidelines for goods comparisons between bio-inspired optimization algorithms. Image taken from [4].

8.3 Bridging Theory and Practice in Bio-inspired Optimization: Application to real-world Problems

The correct design of a bio-inspired algorithm involves the execution of a series of steps in a conscientious and organized manner, both at the time of algorithm development and during subsequent experimentation and application to real-world optimization problems. In [5], a complete tutorial on the design of new bio-inspired algorithms is presented, and in this work, we make a brief introduction to the phases that are necessary for quality research.

In such work, an analysis is conducted from a critical yet constructive point of view, aiming to correct misconceptions and bad methodological habits. Each phase of the analysis includes the prescription of application guidelines and recommendations intended for adoption by the community. These guidelines are intended to promote actionable metaheuristics designed and tested in a principled manner, to achieve valuable research results and ensure their practical use in real-world applications.

Other studies have standardized key optimization concepts, though often focusing narrowly on specific phases or domains. However, this tutorial addresses this gap by offering a comprehensive approach, covering all steps from problem modeling to algorithm validation and implementation. This analysis sheds light on different issues to be solved while designing new bio-inspired algorithms and, to prevent this difficulty, a list of the steps to be performed during the creation of the algorithm is presented, ranging from the early phase of problem modeling to the validation of the developed algorithm, as follows:

- **Problem Modeling and Mathematical Formulation**: Leading by a previous conceptualization of the problem, this phase entails the modeling and mathematical formulation of the optimization problem.
- **Algorithmic Design, Solution Encoding and Search Operators**: The goal of this phase is to design and implement the bio-inspired algorithm. In order to do so, we have to avoid the metaphor and align the design of the algorithm according to the constraints of the problem at hand.
- **Performance Assessment, Comparison and Replicability**: Certain aspects of correct evaluation, applicability, and consistency of the research are studied in this phase. It should be based on good practices as he published in [4] which are resumed in the previous subsection.
- **Algorithmic Deployment for Real-World Applications**: This phase is focused on the study of the deployment of the algorithm in a real environment.

These phases are described in depth in the manuscript, but here we show in Figure 9 a summary of the main recommendations for every phase of the proposed methodology for the design of new bio-inspired optimization algorithms.

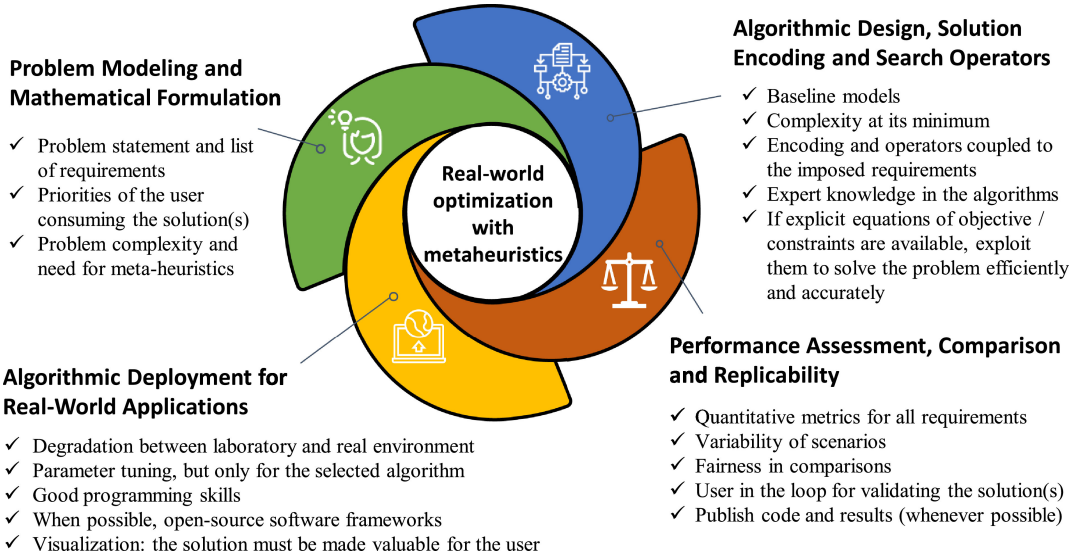


Figure 9: Summary of the recommendations in the four stages of the proposed methodology for the design of new bio-inspired algorithms. Image taken from [5].

9 A Short Recent Literature Analysis: Good Practices, Taxonomies, Overviews, and General Approaches

Since the initial version of this paper in 2020, the field of nature and bio-inspired optimization algorithms has continuously evolved. During these last years, the lack of novelty, and bad comparisons, among others, are described as problems that have to be solved to keep the field in progress. As a result, in Subsection 9.1, we show several studies and guidelines as good practices for designing metaheuristics. At the same time, researchers continue to work on new ways of classifying metaheuristics and publishing general studies on the topic. That is why in Subsection 9.2 we consider, without the intention of being exhaustive with all the studies, a summary of almost a dozen of recent studies about taxonomies [33, 34, 35], overviews [15, 36, 37, 38, 39, 40], and general approaches [41]. For each paper cited in both subsections, we provide its title, year of publication, and a very short summary of its main contents.

9.1 Good Practices for Designing Metaheuristics

The constant evolution of the field leads to a significant issue: the lack of novelty in metaheuristics. However, researchers recognize the need to address this problem and have proposed methods to evaluate the novelty of new algorithms. This section shows different studies and guidelines to measure novelty, to design new metaheuristics, and to perform statistical tests between metaheuristics. We list these approaches as follows:

- **On detecting the novelties in metaphor-based algorithms - 2021 [32]**: This work studies the comparison at the conceptual level using a mathematical formulation based on Markov's chains, and also at the experiment level using the Spearman correlation coefficient between the objective and the population diversity of the algorithms.

- **Similarity in metaheuristics: A gentle step towards a comparison methodology - 2022 [27]**: This paper uses a pool template as a framework for decomposing and analyzing metaheuristics, inspired by another previous work. This template works as a framework for decomposing and analyzing metaheuristics based on these concepts explained in such work: generation method, pool of solutions, archive of solutions, selected pool of solutions, updating mechanism, updated pool, and the archiving and output functions. The authors provide some measures and methodologies to identify their similarities and novelties based on the updating mechanism component, similar to our second taxonomy. They review 15 metaheuristics and their insights confirm that many metaheuristics are special cases of others.

- **Metaheuristics "In the Large" - 2022 [28]**: The objective of this work is to provide a useful tool for researchers. To address the lack of novelty, the authors propose a new infrastructure to support the development, analysis, and comparison of new approaches. This framework is based on (1) the use of algorithm templates for reuse without modification, (2) white box problem descriptions that provide generic support for the injection of domain-specific knowledge, and (3) remotely accessible frameworks, components, and problems. This can be considered as a step towards the improvement of the reproducibility of results.

- **Designing new metaheuristics: Manual versus automatic approaches - 2023 [29]**: This study discusses two methods for the design of new metaheuristics, manual or automatic. Although authors give credit to the manual design of metaheuristics because this development is based on the designer's *intuition* and often involves looking for inspiration in other fields of knowledge, which is a positive aspect. However, they remark that this method could involve finding a good algorithm design in a large set of options through trial and error, possibly leading to eliminating designs that, based on their knowledge, they believe would not work for the problem at hand. For this reason, the authors assure the benefits of automatic design, which seeks to reduce human involvement in the design process by harnessing recent advances in automatic algorithm configuration methods. In this work, several automatic configuration methods and metaheuristic software frameworks from the literature are presented and analyzed, some of them already mentioned in section 6, as steps towards better design of metaheuristics.

- **Research orientation and novelty discriminant for new metaheuristic algorithms - 2024 [26]**: This work proposes a discriminant method based on a mathematical formulation. It provides the division into root and homologous algorithms so that the former represent strongly innovative proposals due to the novelty of their reproduction operators, and the latter does not show any new combinatorial structure about their reproduction operator. This method shows that Harmony

Search, Backtracking Search Optimization Algorithm, Grey Prediction Evolution algorithm, Grey Wolf Optimizer, and Gaining-sharing Knowledge-based algorithm are homologous to several classical algorithms. As a consequence, they develop a research orientation for homologous algorithms to transform the nature metaphor into new structures.

- **Guided learning strategy: A novel update mechanism for metaheuristic algorithms design and improvement - 2024 [30]**: This work provides guidelines for improving the performance of metaheuristics. The authors have developed a strategy for recalling the algorithm's requirements based on the current population. Authors annotate that this novel mechanism is capable of checking that if the algorithm is biased towards exploration, it will shift towards exploitation in subsequent iterations and vice versa. This strategy obtains the dispersion degree of the population by calculating the standard deviation of the historical locations of individuals in recent generations and infers what guidance the algorithm currently needs. This method has been tested with nearly 60 algorithms, validating its effectiveness in improving performance.

- **A Simple statistical test against origin-biased metaheuristics - 2024 [31]**: The authors have developed a test to determine algorithm bias. The test is based on the idea that an unbiased algorithm can choose either direction for one of two different local optima in a function. If there is a difference in behavior between independent runs, then the algorithm is likely biased. Algorithms that are biased in terms of the fitness function can lead to undesired behavior. This paper develops and applies a test to known algorithms, including Grey Wolf Optimizer, Whale Optimization, and Harris Hawk, which fail this test. However, algorithms such as DE, GA, and PSO pass the test. This test is a useful tool to solve the centre-bias problem that has already been studied in [25].

9.2 Latest Metaheuristics based Taxonomies, Overviews, and General Approaches

This section aims to briefly analyze a summary of almost a dozen other taxonomies, overviews, and global approaches developed during these years. Sorted by year of publication, each work is shortly explained to summarize their messages and contribution to the community:

- **A new taxonomy of global optimization algorithms - 2020 [33]**: This work analyzes the four characteristic elements of optimization algorithms: how they initialize, generate, and select solutions, how these solutions are evaluated, and lastly, how these algorithms can be parametrized and controlled. By leveraging these elements, a generalized view of optimization algorithms can be created, just by identifying their specific components. The algorithms, according to those specific components, are classified in a taxonomy based on five categories: hill-climbing, trajectory, population-based, surrogate, or hybrid algorithms. Moreover, the study concludes that most algorithms and algorithm classes have a close connection and share similar components, operators, and a large part of their search strategies, which is the basis for the automated design of new algorithms. Lastly, this work provides a guide for algorithm selection, offering best practices for advanced practitioners when choosing optimization algorithms for new problems.

- **Nature inspired optimization algorithms or simply variations of metaheuristics? - 2021 [15]**: This overview focuses on the study of the frequency of new proposals that are no more than variations of old ones. The authors critique a large set of algorithms based on three criteria: (1) whether there is a physical analogy that follows the metaheuristic, (2) whether most algorithms are duplicates or similarly inspired, and (3) whether the authors propose different techniques based on the same idea. They then specify their criteria for introducing a new metaheuristic.

- **An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges - 2023 [34]**: This taxonomy provides a large classification of metaheuristics based on the number of control parameters of the algorithm. In this work, the authors question the novelty of new proposals and discuss the fact that calling an algorithm new is often based on relatively minor modifications to existing methods. They highlight the limitations of metaheuristics, open challenges, and potential future research directions in the field.

- **Metaheuristics in a nutshell - 2023 [36]**: The purpose of this overview is to define the main terms related to the concept of metaheuristic. The text does not provide an extensive taxonomy, but it clearly distinguishes between two classes of metaheuristics: trajectory and population algorithms. It describes the most well-known algorithms for both classes and for single and multi-objective problems. Finally, quality indicators and statistical analysis are explained as good practices. This overview serves as an introduction to the main concepts, algorithms, and methodologies that every metaheuristics researcher should know.

- **Initialization of metaheuristics: comprehensive review, critical analysis, and research directions - 2023 [35]**: This review addresses a gap in the literature by developing a taxonomy of initialization methods for metaheuristics. This classification is based on the initialization of metaheuristics according to random techniques, learning methods (supervised learning, Markov models, opposition- and diversification-based learning), and other generic methods based on sampling, clustering, and cooperation. The review also examines the initialization of metaheuristics with local search approaches, offers guidance on designing a diverse and informative sequence of initial solutions, and provides insights that will help research in constrained and discrete optimization problems.

- **Metaheuristic optimization algorithms: a comprehensive overview and classification of benchmark test functions - 2024 [37]**: This work focuses on the practical scenario of developing a new metaheuristic. It reviews over 200 mathematical test functions and more than 50 real-world engineering design problems. For each function, it is described its variables and value range. Due to the design of a metaheuristic should be accompanied by a set of experiments, this paper provides researchers with a wide range of options to test the quality of their new developments.

- **A literature review and critical analysis of metaheuristics recently developed - 2024 [38]**: This review focuses on algorithms with titles containing words such as 'new', 'hybrid', or 'improved', in response to the growing trend of nature-based approaches. After analyzing over 100 algorithms, it was found that a significant percentage of these algorithms outperform previous techniques. From the several analyses made in this review, it is noted that most new algorithms are an improved version of some established algorithm, which reveals that the trend is no longer to propose metaheuristics based on new analogies. Moreover, they compare Black Widow Optimization and Coral Reef Optimization, which are considered new frameworks. By analyzing the components of both metaheuristics, authors evident the lack of innovation, as the operators of such algorithms are merely a combination of other evolutionary operators.

- **Metaheuristic optimization algorithms: an overview - 2024 [39]**: This paper focuses on studying the main components and concepts of optimization. More specifically, the overview provides the advantages (agnostic to the problem being solved, gradient independence, global search capability, the capability of dealing with multi-objective optimization problems, balanced exploration and exploitation, configurability and tuning, practical problem-solving, and innovation) and the limitations (absence of global optimality guarantee, convergence speed, parameter tuning, and black-box nature) of metaheuristics. The authors specifically focus on the references used by the algorithms to guide the search, and on how to achieve a good balance between exploration and exploitation. Visual representations accompany the text to illustrate the behavior of a set of metaheuristics.

- **50 years of metaheuristics - 2024 [40]**: This overview traces the last 50 years of the field, starting from the roots of the area to the latest proposals to hybridize metaheuristics with machine learning. The revision encompasses constructive (GRASP and ACO), local search (iterated local search, Tabu search, variable neighborhood search), and population-based heuristics (memetic algorithms, biased random-key genetic algorithms, scatter search, and path relinking). Each category presents its core characteristics and the description of the mentioned algorithms. This review presents metaheuristic frameworks to guide the design of heuristic optimization algorithms during the last 50 years. It discusses the role of the journal in which it is published in introducing solid heuristic papers. This work also recalls the maturity of the field, which leads to solving very complex problems, with a growing number of researchers applying them, as shown in the numerous conferences and related events. Also, they criticize the fragmentation as each group of research usually applies the same methods regardless of the type of problem being solved, the lack of theoretical foundations, the limited analytical understanding of novel proposals, the problem-specific tuning of metaheuristics, the lack of standardized benchmarking protocols and the absence of general guidelines. Several research directions are also annotated for researchers to be applied in the future.

- **Learn to optimize – A brief overview - 2024 [41]**: This paper discusses the concept of Learn to Optimize (L2O) and its application in accelerating the configuration process to obtain a good solver for unseen instances. The studies can be categorized into three main types: training a solver performance prediction model, training a single solver, and training a portfolio of solvers. The first category aims to connect problem instance features with solver performance, resulting in the selection of the best solver, as seen in Automated Algorithm Selection (AAS). The second category, training a single solver, involves finding the best solver for overall performance on the training instances, known as Automatic Algorithm Configuration (AAC). The last category, training a portfolio of solvers, is a more general case of the second category in which a set of solvers is trained, introducing a higher degree of freedom. L2O has achieved importance in general-purpose approaches and other problems like adversarial attacks.

10 Conclusions

Nature and biological organisms have been a source of inspiration for many optimization algorithms. During the last few years, this family of solvers has grown considerably in size, achieving unseen levels of diversity about their source of inspiration. This explosion of literature has made it difficult for the community to appraise the general trajectory followed by the field, which is a necessary step towards identifying research trends and challenges of scientific value and practical impact. Some efforts have been dedicated so far towards classifying the state of the art on nature- and bio-inspired optimization in a taxonomy with well-defined criteria, allowing researchers to classify existing algorithms and newly proposed schemes.

We have reviewed 518 nature- and bio-inspired algorithms and grouped them into two taxonomies. The first taxonomy has considered the source of inspiration, while the second has discriminated algorithms based on their behavior in generating new candidate solutions. We have provided clear descriptions, examples, and an enumeration of the reviewed approaches within each taxonomy category. Our study has critically examined the reviewed literature and found that many algorithms claiming to be inspired by different natural and biological phenomena exhibit algorithmic similarities. Additionally, a significant percentage (24%) of the reviewed proposals have been identified as versions of classical algorithms such as PSO, DE, or GA. These findings shed light on the ongoing debate within the nature- and bio-inspired community regarding the algorithmic contributions of recent advances in the field.

A critical point of reflection associated with this explosion of proposals has been that novel metaphors do not lead to new solvers, and that comparisons undergo serious methodological problems. Although there are increasingly more bio-inspired algorithms, many of them rely on so-claimed novel metaphors that do not create any innovative bio-inspired solvers. In addition, comparisons have been often inadequate, leading to problems of reproducibility and applicability. This problem has captured the interest of other researchers, leading to several papers on various aspects related to bad comparisons and the increasing number of unoriginal proposals, even to the point of not accepting completely new proposals with quality marks. As we have mentioned, we emphasize that in these new algorithms there exists a lack of justification together with the lack of comparison with the state of the art and the lack of real interest in achieving reasonable levels of quality from the perspective of the optimization of well-known problems in recent competitions. Good methodological practices must be followed in forthcoming studies when designing, describing, and comparing new algorithms.

From a positive vision, bio-inspired algorithms have been regularly used in AI and real-world applications. These algorithms hold potential in new scientific avenues, contributing to recent advances in DL evolution [8], the design of large language models (LLM) [627], and more recently, the design and enrichment of GPAIS [628]. GPAIS (including DL evolution and generative AI, such as LLM) are capable of performing tasks beyond those for which they were originally designed. In this context, the paradigm of AI-powered AI, which involves utilizing AI algorithms to enhance other AI systems, assumes paramount importance. In this thrilling era of AI explosion and advancement, we are already witnessing the significant impact of bio-inspired algorithms on the improvement of AI systems through examples like POET [621], EUREKA [624], EvoPrompt [629], and EGANs [630], among others.

In the last update of this report, which is herein released 4 years after its original version, we note that there has been an evolution within the nature and bio-inspired optimization field. There is an excessive use of the biological approach as opposed to the real problem-solving approach to tackle real and complex optimization goals, as those discussed in Section 8.1. This issue needs to be addressed in the future by following guidelines that will allow for the definition of metaheuristics in a way that is appropriate to current challenges. This is important for the constructive design and development of proposals in response to emerging problems. For this reason, the potential impact the emerging problems and GPAIS, population-based metaheuristics as nature and bio-inspired optimization algorithms are poised to shape the future of AI, contributing to the design of continuously emerging AI systems, and serving as an inspiration for the new era of innovation and progress in AI.

References

[1] Molina D, Poyatos J, Ser JD, García S, Hussain A, Herrera F. Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. Cognitive Computation. 2020;12; p. 897–939.

[2] Molina Cabrera D, Poyatos Amador J, Osaba Icedo E, Del Ser Lorente J, Herrera Triguero F. Nature-and bio-inspired optimization: the good, the bad, the ugly and the hopeful. DYNA Ingeniería e Industria. 2022;97(2); p. 114–117.

[3] Del Ser J, Osaba E, Molina D, Yang XS, Salcedo-Sanz S, Camacho D, et al. Bio-inspired computation: Where we stand and what's next. Swarm and Evolutionary Computation. 2019;48; p. 220–250.

[4] LaTorre A, Molina D, Osaba E, Poyatos J, Del Ser J, Herrera F. A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. Swarm and Evolutionary Computation. 2021;67; p. 100973.

[5] Osaba E, Villar-Rodriguez E, Del Ser J, Nebro AJ, Molina D, LaTorre A, et al. A Tutorial On the design, experimentation and application of metaheuristic algorithms to real-World optimization problems. Swarm and Evolutionary Computation. 2021;64; p. 100888.

[6] Pintea CM. Bio-inspired Computing. In: Advances in Bio-inspired Computing for Combinatorial Optimization Problems. Springer Berlin Heidelberg; 2014. p. 3–19.

[7] Mahdavi S, Shiri ME, Rahnamayan S. Metaheuristics in large-scale global continues optimization: A survey. Information Sciences. 2015;295; p. 407–428.

[8] Martinez AD, Del Ser J, Villar-Rodriguez E, Osaba E, Poyatos J, Tabik S, et al. Lights and shadows in Evolutionary Deep Learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges. Information Fusion. 2021;67; p. 161–194.

[9] Sörensen K. Metaheuristics–the metaphor exposed. International Transactions In Operational Research. 2015;22(1); p. 3–18.

[10] Fister Jr I, Mlakar U, Brest J, Fister I. A new population-based nature-inspired algorithm every month: is the current era coming to the end. In: Proceedings of the 3rd Student Computer Science Research Conference; 2016. p. 33–37.

[11] Weyland D. A critical analysis of the harmony search algorithm – how not to solve sudoku. Operations Research Perspectives. 2015;2; p. 97–105.

[12] Saka MP, Hasançebi O, Geem ZW. Metaheuristics in structural optimization and discussions on harmony search algorithm. Swarm and Evolutionary Computation. 2016;28; p. 88–97.

[13] Piotrowski AP, Napiorkowski JJ, Rowinski PM. How novel is the "novel" black hole optimization approach? Information Sciences. 2014;267; p. 191–200.

[14] Camacho Villalón CL, Stützle T, Dorigo M. Grey Wolf, Firefly and Bat Algorithms: Three Widespread Algorithms that Do Not Contain Any Novelty. In: International Conference on Swarm Intelligence. vol. 12421; 2020. p. 121–133.

[15] Tzanetos A, Dounias G. Nature inspired optimization algorithms or simply variations of metaheuristics? Artificial Intelligence Review. 2021;54(3); p. 1841–1862.

[16] Aranha C, Camacho Villalón CL, Campelo F, Dorigo M, Ruiz R, Sevaux M, et al. Metaphor-based metaheuristics, a call for action: the elephant in the room. Swarm Intelligence. 2022;16; p. 1–6.

[17] Piotrowski AP, Napiorkowski JJ. Some metaheuristics should be simplified. Information Sciences. 2018;427; p. 32–62.

[18] Camacho-Villalón CL, Dorigo M, Stützle T. Why the Intelligent Water Drops Cannot Be Considered as a Novel Algorithm. In: Swarm Intelligence; 2018. p. 302–314.

[19] Camacho-Villalón CL, Dorigo M, Stützle T. The intelligent water drops algorithm: why it cannot be considered a novel algorithm: A brief discussion on the use of metaphors in optimization. Swarm Intelligence. 2019;13; p. 173–192.

[20] Camacho-Villalón CL, Dorigo M, Stützle T. An analysis of why cuckoo search does not bring any novel ideas to optimization. Computers & Operations Research. 2022;142; p. 105747.

[21] Kudela J. A critical problem in benchmarking and analysis of evolutionary computation methods. Nature Machine Intelligence. 2022;4(12); p. 1238–1245.

[22] Campelo F, Aranha C. Lessons from the Evolutionary Computation Bestiary. Artificial Life. 2023;29(4); p. 421–432.

[23] Tzanetos A. Does the Field of Nature-Inspired Computing Contribute to Achieving Lifelike Features? Artificial Life. 2023;29(4); p. 487–511.

[24] Camacho-Villalón CL, Dorigo M, Stützle T. Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors. International Transactions in Operational Research. 2023;30(6); p. 2945–2971.

[25] Kudela J. The Evolutionary Computation Methods No One Should Use; 2023.

[26] Hu Z, Zhang Q, Wang Y, Su Q, Xiong Z. Research orientation and novelty discriminant for new metaheuristic algorithms. Applied Soft Computing. 2024;157; p. 111521.

[27] de Armas J, Lalla-Ruiz E, Tilahun SL, Voß S. Similarity in metaheuristics: A gentle step towards a comparison methodology. Natural Computing. 2022;21; p. 265–287.

[28] Swan J, Adriaensen S, Brownlee AEI, Hammond K, Johnson CG, Kheiri A, et al. Metaheuristics "In the Large". European Journal of Operational Research. 2022;297(2); p. 393–406.

[29] Camacho-Villalón CL, Stützle T, Dorigo M. Designing New Metaheuristics: Manual Versus Automatic Approaches. Intelligent Computing. 2023;2; p. 0048.

[30] Jia H, Lu C. Guided learning strategy: A novel update mechanism for metaheuristic algorithms design and improvement. Knowledge-Based Systems. 2024;286; p. 111402.

[31] Walden A, Buzdalov M. A Simple Statistical Test Against Origin-Biased Metaheuristics. In: International Conference on the Applications of Evolutionary Computation; 2024. p. 322–337.

[32] Fister I, Fister I, Iglesias A, Galvez A. On detecting the novelties in metaphor-based algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion; 2021. p. 71–72.

[33] Stork J, Eiben AE, Bartz-Beielstein T. A new taxonomy of global optimization algorithms. Natural Computing. 2020;(21); p. 219–242.

[34] Rajwar K, Deep K, Das S. An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges. Artificial Intelligence Review. 2023;56; p. 13187–13257.

[35] Sarhani M, Voß S, Jovanovic R. Initialization of metaheuristics: comprehensive review, critical analysis, and research directions. International Transactions in Operational Research. 2023;30(6); p. 3361–3397.

[36] Ferrer J, Delgado-Pérez P. Metaheuristics in a Nutshell. In: Optimising the Software Development Process with Artificial Intelligence. Springer; 2023. p. 279–307.

[37] Sharma P, Raju S. Metaheuristic optimization algorithms: A comprehensive overview and classification of benchmark test functions. Soft Computing. 2024;28(4); p. 3123–3186.

[38] Velasco L, Guerrero H, Hospitaler A. A literature review and critical analysis of metaheuristics recently developed. Archives of Computational Methods in Engineering. 2024;31; p. 125–146.

[39] Brahim B, Kobayashi M, Al Ali M, Khatir T, Elmeliani MEAE. Metaheuristic Optimization Algorithms: an overview. HCMCOU Journal of Science–Advances in Computational Structures. 2024;14(1); p. 1–28.

[40] Martí R, Sevaux M, Sörensen K. 50 years of metaheuristics. In Press European Journal of Operational Research. 2024;DOI: 10.1016/j.ejor.2024.04.004.

[41] Tang K, Yao X. Learn to Optimize-A Brief Overview. National Science Review. 2024;p. 1–10.

[42] Kar AK. Bio inspired computing – A review of algorithms and scope of applications. Expert Systems with Applications. 2016;59; p. 20–32.

[43] Xiong N, Molina D, Ortiz ML, Herrera F. A walk into metaheuristics for engineering optimization: principles, methods and recent trends. International Journal of Computational Intelligence Systems. 2015;8(4); p. 606–636.

[44] Molina D, LaTorre A, Herrera F. An Insight into Bio-inspired and Evolutionary Algorithms for Global Optimization: Review, Analysis, and Lessons Learnt over a Decade of Competitions. Cognitive Computation. 2018;10(4); p. 517–544.

[45] Zavala GR, Nebro AJ, Luna F, Coello Coello CA. A survey of multi-objective metaheuristics applied to structural optimization. Structural and Multidisciplinary Optimization. 2014;49(4); p. 537–558.

[46] Yang XS, Chien SF, Ting TO. Bio-Inspired Computation in Telecommunications. Morgan Kaufmann; 2015. p. 1–21.

[47] Beni G, Wang J. Swarm Intelligence in Cellular Robotic Systems. In: Robots and Biological Systems: Towards a New Bionics?; 1993. p. 703–712.

[48] Fong S. Opportunities and Challenges of Integrating Bio-Inspired Optimization and Data Mining Algorithms. In: Swarm Intelligence and Bio-Inspired Computation. Elsevier; 2013. p. 385–402.

[49] Del Ser J, Osaba E, Sanchez-Medina JJ, Fister I. Bioinspired computational intelligence and transportation systems: a long road ahead. IEEE Transactions on Intelligent Transportation Systems. 2019;21(2); p. 466–495.

[50] del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez J, Harley RG. Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. IEEE Transactions on Evolutionary Computation. 2008;12(2); p. 171–195.

[51] Dressler F, Akan OB. A survey on bio-inspired networking. Computer Networks. 2010;54(6); p. 881 – 900.

[52] José-García A, Gómez-Flores W. Automatic clustering using nature-inspired metaheuristics: A survey. Applied Soft Computing. 2016;41; p. 192 – 213.

[53] Alsalibi B, Venkat I, Subramanian KG, Lutfi SL, Wilde PD. The Impact of Bio-Inspired Approaches Toward the Advancement of Face Recognition. ACM Computing Surveys. 2015;48(5); p. 1–33.

[54] García-Godoy MJ, López-Camacho E, García-Nieto J, Del Ser J, Nebro AJ, Aldana-Montes JF. Bio-inspired optimization for the molecular docking problem: State of the art, recent results and perspectives. Applied Soft Computing. 2019;79; p. 30–45.

[55] Kolias C, Kambourakis G, Maragoudakis M. Swarm intelligence in intrusion detection: A survey. Computers and Security. 2011;30(8); p. 625–642.

[56] Banks A, Vincent J, Anyakoha C. A review of particle swarm optimization. Part I: background and development. Natural Computing. 2007;6(4); p. 467–484.

[57] Neri F, Tirronen V. Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review. 2010;33; p. 61–106.

[58] Das S, Suganthan PN. Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation. 2011;15(1); p. 4–31.

[59] Das S, Mullick SS, Suganthan PN. Recent advances in differential evolution – An updated survey. Swarm and Evolutionary Computation. 2016;27; p. 1–30.

[60] Karaboga D, Gorkemli B, Ozturk C, Karaboga N. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial Intelligence Review. 2014;42; p. 21–57.

[61] Bitam S, Batouche M, Talbi E. A survey on bee colony algorithms. In: 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW); 2010. p. 1–8.

[62] Das S, Biswas A, Dasgupta S, Abraham A. Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications. In: Foundations of Computational Intelligence Volume 3: Global Optimization. Springer Berlin Heidelberg; 2009. p. 23–55.

[63] Yang XS, He X. Bat Algorithm: Literature Review and Applications. International Journal of Bio-Inspired Computation. 2013;5(3); p. 141–149.

[64] Bonabeau E, Dorigo M, Théraulaz G. Swarm intelligence: from natural to artificial systems. Oxford University press; 1999.

[65] Yang XS. Nature-Inspired Optimization Algorithms. Elsevier; 2014.

[66] Das S, Abraham A, Konar A. Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives. In: Advances of Computational Intelligence in Industrial Systems. Springer Berlin Heidelberg; 2008. p. 1–38.

[67] Elbeltagi E, Hegazy T, Grierson D. Comparison among five evolutionary-based optimization algorithms. Advanced Engineering Informatics. 2005;19(1); p. 43–53.

[68] Pazhaniraja N, Paul PV, Roja G, Shanmugapriya K, Sonali B. A study on recent bio-inspired optimization algorithms. In: 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN); 2017. p. 1–6.

[69] Krömer P, Platoš J, Snášel V. Nature-Inspired Meta-Heuristics on Modern GPUs: State of the Art and Brief Survey of Selected Algorithms. International Journal of Parallel Programming. 2014;42(5); p. 681–709.

[70] Piotrowski AP, Napiorkowski M, Napiorkowski JJ, Rowinski PM. Swarm Intelligence and Evolutionary Algorithms: performance versus speed. Information Sciencies. 2017;384; p. 34–85.

[71] El-Abd M. Performance assessment of foraging algorithms vs. evolutionary algorithms. Information Sciences. 2012;182(1); p. 243–263.

[72] Chouikhi N, Ammar B, Hussain A, Alimi AM. Bi-level multi-objective evolution of a Multi-Layered Echo-State Network Autoencoder for data representations. Neurocomputing. 2019;341; p. 195–211.

[73] Chouikhi N, Ammar B, Rokbani N, Alimi AM. PSO-based analysis of Echo State Network parameters for time series forecasting. Applied Soft Computing. 2017;55; p. 211–225.

[74] Fister jr I, Yang XS, Fister I, Brest J, Fister D. A Brief Review of Nature-Inspired Algorithms for Optimization. Elektrotehniski Vestnik. 2013;80(3); p. 1–7.

[75] Baskaran A, Balaji N, Basha S, Vengattaraman T. A survey of nature inspired algorithms. International Journal of Applied Engineering Research. 2015;10; p. 19313–19324.

[76] Rajakumar R, Dhavachelvan P, Vengattaraman T. A survey on nature inspired meta-heuristic algorithms with its domain specifications. In: 2016 International Conference on Communication and Electronics Systems (ICCES); 2016. p. 1–6.

[77] Kumar Kar A. Bio inspired computing – A review of algorithms and scope of applications. Expert Systems With Applications. 2016;59; p. 20–32.

[78] Chu X, Wu T, Weir JD, Shi Y, Niu B, Li L. Learning–interaction–diversification framework for swarm intelligence optimizers: a unified perspective. Neural Computing and Applications. 2020;(32); p. 1789—-1809.

[79] Kirkpatrick S, Gelatt CD, P VM. Optimization by Simulated Annealing. Science. 1989;220(4598); p. 671–680.

[80] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science; 1995. p. 39–43.

[81] Braik MS. Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. Expert Systems with Applications. 2021;174; p. 114685.

[82] Adham M, Bentley P. An Artificial Ecosystem Algorithm applied to static and Dynamic Travelling Salesman Problems. In: IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - IEEE ICES: 2014 IEEE International Conference on Evolvable Systems, Proceedings; 2015. p. 149–156.

[83] Zhao W, Wang L, Zhang Z. Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. Neural Computing and Applications. 2020;32; p. 9383–9425.

[84] Huang G. Artificial infectious disease optimization: A SEIQR epidemic dynamic model-based function optimization algorithm. Swarm and Evolutionary Computation. 2016;27; p. 31–67.

[85] Farasat A, Menhaj MB, Mansouri T, Sadeghi Moghadamd MR. ARO: A new model-free optimization algorithm inspired from asexual reproduction. Applied Soft Computing. 2010;10(4); p. 1284–1292.

[86] Simon D. Biogeography-Based Optimization. IEEE Transactions on Evolutionary Computation. 2008;12(6); p. 702–713.

[87] Askarzadeh A. Bird mating optimizer: An optimization algorithm inspired by bird mating strategies. Communications in Nonlinear Science and Numerical Simulation. 2014;19(4); p. 1213 – 1228.

[88] Zhang X, Sun B, Mei T, Wang R. Post-disaster restoration based on fuzzy preference relation and Bean Optimization Algorithm. In: 2010 IEEE Youth Conference on Information, Computing and Telecommunications; 2010. p. 271–274.

[89] Yuan Y, Shen Q, Wang S, Ren J, Yang D, Yang Q, et al. Coronavirus mask protection algorithm: A new bio-inspired optimization algorithm and its applications. Journal of Bionic Engineering. 2023;20; p. 1747–1765.

[90] Khalid AM, Hosny KM, Mirjalili S. COVIDOA: a novel evolutionary optimization algorithm based on coronavirus disease replication lifecycle. Neural Computing and Applications. 2022;34(24); p. 22465–22492.

[91] Salcedo-Sanz S, Del Ser J, Landa-Torres I, Gil-López S, Portilla-Figueras J. The coral reefs optimization algorithm: a novel metaheuristic for efficiently solving optimization problems. The Scientific World Journal. 2014;2014; p. 739768.

[92] Greensmith J, Aickelin U, Cayzer S. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In: International Conference on Artificial Immune Systems; 2005. p. 153–167.

[93] Price K, Storn R. A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization. 1997;11(4); p. 341–359.

[94] Zheng YJ, Ling HF, Xue JY. Ecogeography-based optimization: Enhancing biogeography-based optimization with ecogeographic barriers and differentiations. Computers and Operations Research. 2014;50; p. 115–127.

[95] Parpinelli RS, Lopes HS. An eco-inspired evolutionary algorithm applied to numerical optimization. In: 2011 Third World Congress on Nature and Biologically Inspired Computing; 2011. p. 466–471.

[96] Wang GG, Deb S, dos Santos Coelho L. Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems. IJBIC. 2018;12(1); p. 1–22.

[97] Beyer HG, Schwefel HP. Evolution strategies – A comprehensive introduction. Natural Computing. 2002;1; p. 3–52.

[98] Man KF, Tang KS, Kwong S. Genetic algorithms: concepts and applications [in engineering design]. IEEE Transactions on Industrial Electronics. 1996;43(5); p. 519–534.

[99] Ferreira C. Gene Expression Programming in Problem Solving. In: Soft Computing and Industry: Recent Applications. Springer London; 2002. p. 635–653.

[100] Ye Z, Ma L, Chen H. A hybrid rice optimization algorithm. In: 2016 11th International Conference on Computer Science Education (ICCSE); 2016. p. 169–174.

[101] Cortés P, García JM, Onieva L, Muñuzuri J, Guadix J. Viral System to Solve Optimization Problems: An Immune-Inspired Computational Intelligence Approach. In: Artificial Immune Systems; 2008. p. 83–94.

[102] Tayeb FBS, Bessedik M, Benbouzid M, Cheurfi H, Blizak A. Research on permutation flow-shop scheduling problem based on improved genetic immune algorithm with vaccinated offspring. Procedia Computer Science. 2017;112; p. 427–436.

[103] Mehrabian AR, Lucas C. A novel numerical optimization algorithm inspired from weed colonization. Ecological Informatics. 2006;1(4); p. 355–366.

[104] Abbass HA. MBO: marriage in honey bees optimization-a Haplometrosis polygynous swarming approach. In: Proceedings of the 2001 IEEE Congress on Evolutionary Computation. vol. 1; 2001. p. 207–214.

[105] Bidar M, Kanan HR, Mouhoub M, Sadaoui S. Mushroom Reproduction Optimization (MRO): A Novel Nature-Inspired Evolutionary Algorithm. In: 2018 IEEE Congress on Evolutionary Computation (CEC); 2018. p. 1–10.

[106] Sung Hoon Jung. Queen-bee evolution for genetic algorithms. Electronics Letters. 2003;39(6); p. 575–576.

[107] Anandaraman C, Sankar AVM, Natarajan R. A new evolutionary algorithm based on bacterial evolution and its application for scheduling a flexible manufacturing system. Jurnal Teknik Industri. 2012;14(1); p. 1–12.

[108] Taherdangkoo M, Yazdi M, Bagheri MH. Stem Cells Optimization Algorithm. In: Bio-Inspired Computing and Applications; 2012. p. 394–403.

[109] Nara K, Takeyama T, Hyungchul Kim. A new evolutionary algorithm based on sheep flocks heredity model and its application to scheduling problem. In: IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics. vol. 6; 1999. p. 503–508.

[110] Pattnaik SS, Bakwad KM, Sohi BS, Ratho RK, Devi S. Swine Influenza Models Based Optimization (SIMBO). Applied Soft Computing. 2013;13(1); p. 628–653.

[111] Zelinka I. SOMA — Self-Organizing Migrating Algorithm. In: New Optimization Techniques in Engineering. Springer Berlin Heidelberg; 2004. p. 167–217.

[112] Zhang H, Zhang Y, Niu Y, He K, Wang Y. T Cell Immune Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications. IEEE Access. 2023;11; p. 95545–95566.

[113] Puris A, Bello R, Molina D, Herrera F. Variable mesh optimization for continuous optimization problems. Soft Computing. 2012;16(3); p. 511–525.

[114] Jaderyan M, Khotanlou H. Virulence Optimization Algorithm. Applied Soft Computing. 2016;43; p. 596–618.

[115] Dorigo M, Maniezzo V, Colorni A. The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). 1996;26(1); p. 29–41.

[116] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization. 2007;39(3); p. 459–471.

[117] Yang XS. Firefly Algorithms for Multimodal Optimization. In: Stochastic Algorithms: Foundations and Applications; 2009. p. 169–178.

[118] Saremi S, Mirjalili S, Lewis A. Grasshopper Optimisation Algorithm: Theory and application. Advances in Engineering Software. 2017;105; p. 30–47.

[119] Brabazon A, McGarraghy S. Foraging-Inspired Optimisation Algorithms. Natural Computing Series, Springer; 2018.

[120] Uymaz SA, Tezel G, Yel E. Artificial algae algorithm (AAA) for nonlinear global optimization. Applied Soft Computing. 2014;31; p. 153–171.

[121] Muñoz MA, López JA, Caicedo E. An artificial beehive algorithm for continuous optimization. International Journal of Intelligent Systems. 2009;24(11); p. 1080–1093.

[122] Naderi B, Khalili M, Khamseh AA. Mathematical models and a hunting search algorithm for the no-wait flowshop scheduling with parallel machines. International Journal of Production Research. 2014;52(9); p. 2667–2681.

[123] Odili JB, Mohmad Kahar MN. Solving the Traveling Salesman's Problem Using the African Buffalo Optimization. Computational Intelligence and Neuroscience. 2016;2016; p. 1510256.

[124] Almonacid B, Soto R. Andean Condor Algorithm for cell formation problems. Natural Computing. 2019;18(2); p. 351–381.

[125] Lamy JB. Artificial Feeding Birds (AFB): a new metaheuristic inspired by the behavior of pigeons. In: Advances in nature-inspired computing and applications; 2019. p. 43–60.

[126] Zhao W, Wang L, Mirjalili S. Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. Computer Methods in Applied Mechanics and Engineering. 2022;388; p. 114194.

[127] Zitouni F, Harous S, Belkeram A, Hammou LEB. The archerfish hunting optimizer: A novel metaheuristic algorithm for global optimization. Arabian Journal for Science and Engineering. 2022;47(2); p. 2513–2553.

[128] Li X, Zhang J, Yin M. Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Computing and Applications. 2014;24(7); p. 1867–1877.

[129] Liu R, Zhou N, Yao Y, Yu F. An aphid inspired metaheuristic optimization algorithm and its application to engineering. Scientific Reports. 2022;12; p. 18064.

[130] Mirjalili S. The Ant Lion Optimizer. Advances in Engineering Software. 2015;83; p. 80–98.

[131] Abualigah L, Yousri D, Abd Elaziz M, Ewees AA, Al-qaness MAA, Gandomi AH. Aquila Optimizer: A novel meta-heuristic optimization algorithm. Computers & Industrial Engineering. 2021;157; p. 107250.

[132] Pook MF, Ramlan EI. The Anglerfish algorithm: a derivation of randomized incremental construction technique for solving the traveling salesman problem. Evolutionary Intelligence. 2019;12; p. 11–20.

[133] Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH. The Arithmetic Optimization Algorithm. Computer Methods in Applied Mechanics and Engineering. 2021;376; p. 113609.

[134] Wang L, Cao Q, Zhang Z, Mirjalili S, Zhao W. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. Engineering Applications of Artificial Intelligence. 2022;114; p. 105082.

[135] Chen T, Pang L, Jiang Du, Liu Z, Zhang L. Artificial Searching Swarm Algorithm for solving constrained optimization problems. In: 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems. vol. 1; 2009. p. 562–565.

[136] Chen T, Wang Y, Li J. Artificial tribe algorithm and its performance analysis. Journal of Software. 2012;7(3); p. 651–656.

[137] Subramanian C, Sekar ASS, Subramanian K. A New Engineering Optimization Method African Wild Dog Algorithm. International Journal of Soft Computing. 2013;8(3); p. 163–170.

[138] Mohapatra S, Mohapatra P. American zebra optimization algorithm for global optimization problems. Scientific Reports. 2023;13; p. 5211.

[139] Alsattar HA, Zaidan AA, Zaidan BB. Novel meta-heuristic bald eagle search optimisation algorithm. Artificial Intelligence Review. 2019;(53); p. 2237–2264.

[140] Pham DT, Ghanbarzadeh A, Koç E, Otri S, Rahim S, Zaidi M. The Bees Algorithm — A Novel Tool for Complex Optimisation Problems. In: Intelligent Production Machines and Systems; 2006. p. 454–459.

[141] Comellas F, Martinez-Navarro J. Bumblebees: A Multiagent Combinatorial Optimization Algorithm Inspired by Social Insect Behaviour. In: Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation; 2009. p. 811–814.

[142] Kazikova A, Pluhacek M, Senkerik R, Viktorin A. Proposal of a new swarm optimization method inspired in bison behavior. Advances in Intelligent Systems and Computing. 2019;837; p. 146–156.

[143] Häckel S, Dippold P. The Bee Colony-inspired Algorithm (BCiA): A Two-stage Approach for Solving the Vehicle Routing Problem with Time Windows. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation; 2009. p. 25–32.

[144] Teodorović D, Dell'Orco M. Bee colony optimization - A cooperative learning approach to complex transportation problems. Advanced OR and AI Methods in Transportation. 2005;51; p. 51–60.

[145] Niu B, Wang H. Bacterial Colony Optimization: Principles and Foundations. In: Emerging Intelligent Computing Technology and Applications; 2012. p. 501–506.

[146] Müller SD, Marchetto J, Airaghi S, Koumoutsakos P. Optimization Based on Bacterial Chemotaxis. IEEE Transactions On Evolutionary Computation. 2002;6(1); p. 16–29.

[147] Dutta T, Bhattacharyya S, Dey S, Platos J. Border Collie Optimization. IEEE Access. 2020;8; p. 109177–109197.

[148] Lui Y, Passino KM. Biomimicry of Social Foraging Bacteria for Distributed Optimization: Models, Principles, and Emergent Behaviors. Journal of Optimization Theory and Applications. 2002;115(3); p. 603–628.

[149] Chen T, Tsai P, Chu S, Pan J. A Novel Optimization Approach: Bacterial-GA Foraging. In: Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007); 2007. p. 391–391.

[150] Wedde HF, Farooq M, Zhang Y. BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior. In: Ant Colony Optimization and Swarm Intelligence, Proceeding; 2004. p. 83–94.

[151] Bitam S, Zeadally S, Mellouk A. Fog computing job scheduling optimization based on bees swarm. Enterprise Information Systems. 2018;12(4); p. 373–397.

[152] Malakooti B, Kim H, Sheikh S. Bat intelligence search with application to multi-objective multiprocessor scheduling optimization. International Journal of Advanced Manufacturing Technology. 2012;60(9-12); p. 1071–1086.

[153] Yang XS. A New Metaheuristic Bat-Inspired Algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer; 2010. p. 65–74.

[154] Zhang Q, Wang R, Yang J, Lewis A, Chiclana F, Yang S. Biology migration algorithm: a new nature-inspired heuristic methodology for global optimization. Soft Computing. 2019;23(16); p. 7333–7358.

[155] Sulaiman MH, Mustaffa Z, Saari MM, Daniyal H, Mohamad AJ, Othman MR, et al. Barnacles Mating Optimizer Algorithm for Optimization. In: Proceedings of the 10th National Technical Seminar on Underwater System Technology 2018; 2019. p. 211–218.

[156] Taherdangkoo M, Shirzadi MH, Yazdi M, Bagher MH. A robust clustering method based on blind, naked mole-rats (BNMR) algorithm. Swarm and Evolutionary Computation. 2013;10; p. 1–11.

[157] Kumar A, Misra RK, Singh D. Butterfly optimizer. In: 2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI); 2015. p. 1–6.

[158] Das AK, Pratihar DK. A new bonobo optimizer (BO) for real-parameter optimization. In: 2019 IEEE Region 10 Symposium (TENSYMP); 2019. p. 108–113.

[159] Findik O. Bull optimization algorithm based on genetic operators for continuous optimization problems. Turkish Journal of Electrical Engineering & Computer Sciences. 2015;23; p. 2225–2239.

[160] Sato T, Hagiwara M. Bee System: Finding solution by a concentrated search. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. vol. 4; 1997. p. 3954–3959.

[161] Lucic P, Teodorovic D. Transportation modeling: an artificial life approach. In: Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2002); 2002. p. 216–223.

[162] Meng XB, Gao XZ, Lu L, Liu Y, Zhang H. A new bio-inspired optimisation algorithm: Bird Swarm Algorithm. Journal of Experimental and Theoretical Artificial Intelligence. 2016;28(4); p. 673–687.

[163] Akbari R, Mohammadi A, Ziarati K. A novel bee swarm optimization algorithm for numerical function optimization. Communications in Nonlinear Science and Numerical Simulation. 2010;15(10); p. 3142–3155.

[164] de Oliveira DR, Parpinelli RS, Lopes HS. Bioluminescent Swarm Optimization Algorithm. In: Evolutionary Algorithms. IntechOpen; 2011. .

[165] Wang L, Zhang Q, He X, Yang S, Jiang S, Dong Y. Biological survival optimization algorithm with its engineering and neural network applications. Soft Computing. 2023;27(10); p. 6437–6463.

[166] Drias H, Sadeg S, Yahi S. Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem. In: Computational Intelligence and Bioinspired Systems; 2005. p. 318–325.

[167] Arshaghi A, Ashourian M, Ghabeli L. Buzzard optimization algorithm: a nature-inspired metaheuristic algorithm. Majlesi Journal of Electrical Engineering. 2019;13(3); p. 83–98.

[168] Hayyolalam V, Pourhaji Kazem AA. Black Widow Optimization Algorithm: A novel meta-heuristic approach for solving engineering optimization problems. Engineering Applications of Artificial Intelligence. 2020;87; p. 103249.

[169] Zhong C, Li G, Meng Z. Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. Knowledge-Based Systems. 2022;251; p. 109215.

[170] K SR, Panwar L, Panigrahi BK, Kumar R. Binary whale optimization algorithm: a new metaheuristic approach for profit-based unit commitment problems in competitive electricity markets. Engineering Optimization. 2019;51(3); p. 369–389.

[171] Cuevas E, González M, Zaldivar D, Pérez-Cisneros M, García G. An algorithm for global optimization inspired by collective animal behavior. Discrete Dynamics in Nature and Society. 2012;2012; p. 638275.

[172] Klein CE, Mariani VC, Coelho LDS. Cheetah based optimization algorithm: A novel swarm intelligence paradigm. In: ESANN 2018 - Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning; 2018. p. 685–690.

[173] Shiqin Y, Jianjun J, Guangxing Y. Improved binary particle swarm optimization using catfish effect for feature selection. Expert Systems with Applications. 2011;38(10); p. 12699–12707.

[174] Canayaz M, Karci A. Cricket behaviour-based evolutionary computation technique in solving engineering optimization problems. Applied Intelligence. 2016;44(2); p. 362–376.

[175] Pierezan J, Maidl G, Massashi Yamao E, dos Santos Coelho L, Cocco Mariani V. Cultural coyote optimization algorithm applied to a heavy duty gas turbine operation. Energy Conversion and Management. 2019;199; p. 111932.

[176] Rizk-Allah RM, Hassanien AE, Bhattacharyya S. Chaotic crow search algorithm for fractional optimization problems. Applied Soft Computing. 2018;71; p. 1161–1175.

[177] Sayed GI, Tharwat A, Hassanien AE. Chaotic dragonfly algorithm: an improved metaheuristic algorithm for feature selection. Applied Intelligence. 2019;49; p. 188–205.

[178] Eesa Sabry A, Adbulazeez Brifcani AM, Orman Z. Cuttlefish Algorithm – A Novel Bio-Inspired Optimization Algorithm. International Journal of Scientific and Engineering Research. 2013;4(9); p. 1978–1986.

[179] Iordache S. A Hierarchical Cooperative Evolutionary Algorithm. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation; 2010. p. 225–232.

[180] Al-Obaidi A, Abdullah H, Othman Z. Camel Herds Algorithm: a New Swarm Intelligent Algorithm to Solve Optimization Problems. International Journal on Perceptive and Cognitive Computing. 2017;3(1); p. 1–5.

[181] Khishe M, Mosavi MR. Chimp optimization algorithm. Expert Systems with Applications. 2020;149; p. 113338.

[182] Rajabioun R. Cuckoo Optimization Algorithm. Applied Soft Computing. 2011;11(8); p. 5508–5518.

[183] Ibrahim MK, Salim Ali R. Novel Optimization Algorithm Inspired by Camel Traveling Behavior. Iraq Journal Electrical and Electronic Engineering. 2016;12(2); p. 167–177.

[184] Pierezan J, Dos Santos Coelho L. Coyote Optimization Algorithm: A New Metaheuristic for Global Optimization Problems. In: 2018 IEEE Congress on Evolutionary Computation (CEC); 2018. p. 1–8.

[185] Naruei I, Keynia F. A new optimization method based on COOT bird natural life model. Expert Systems with Applications. 2021;183; p. 115352.

[186] Dehghani M, Montazeri Z, Trojovská E, Trojovskỳ P. Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Systems. 2023;259; p. 110011.

[187] Abdel-Basset M, Mohamed R, Abouhawwash M. Crested Porcupine Optimizer: A new nature-inspired metaheuristic. Knowledge-Based Systems. 2024;284; p. 111257.

[188] Yang X, Deb S. Cuckoo Search via Lévy flights. In: 2009 World Congress on Nature Biologically Inspired Computing (NaBIC); 2009. p. 210–214.

[189] Askarzadeh A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers and Structures. 2016;169; p. 1–12.

[190] Qais MH, Hasanien HM, Turky RA, Alghuwainem S, Tostado-Véliz M, Jurado F. Circle search algorithm: A geometry-based metaheuristic optimization algorithm. Mathematics. 2022;10(10); p. 1626.

[191] Chu SC, Tsai Pw, Pan JS. Cat Swarm Optimization. In: PRICAI 2006: Trends in Artificial Intelligence; 2006. p. 854–858.

[192] Meng X, Liu Y, Gao X, Zhang H. A New Bio-inspired Algorithm: Chicken Swarm Optimization. In: Advances in Swarm Intelligence; 2014. p. 86–94.

[193] Mirjalili S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications. 2016;27(4); p. 1053–1073.

[194] Bhardwaj S, Kim DS. Dragonfly-based swarm system model for node identification in ultra-reliable low-latency communication. Neural Computing and Applications. 2020;33; p. 1837–1880.

[195] Kaveh A, Farhoudi N. A new optimization method: Dolphin echolocation. Advances in Engineering Software. 2013;59; p. 53–70.

[196] Ahmadi B, Giraldo JS, Hoogsteen G. Dynamic Hunting Leadership optimization: Algorithm and applications. Journal of Computational Science. 2023;69; p. 102010.

[197] Brammya G, Praveena S, Ninu Preetha NS, Ramya R, Rajakumar BR, Binu D. Deer Hunting Optimization Algorithm: A New Nature-Inspired Meta-heuristic Paradigm. The Computer Journal. 2019;p. 1–20.

[198] Agushaka JO, Ezugwu AE, Abualigah L. Dwarf Mongoose Optimization Algorithm. Computer Methods in Applied Mechanics and Engineering. 2022;391; p. 114570.

[199] Zhao S, Zhang T, Ma S, Chen M. Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications. Engineering Applications of Artificial Intelligence. 2022;114; p. 105075.

[200] Bairwa AK, Joshi S, Singh D. Dingo optimizer: a nature-inspired metaheuristic approach for engineering problems. Mathematical Problems in Engineering. 2021;2021; p. 2571863.

[201] Shiqin Y, Jianjun J, Guangxing Y. A Dolphin Partner Optimization. In: 2009 WRI Global Congress on Intelligent Systems. vol. 1; 2009. p. 124–128.

[202] Dehghani M, Mardaneh M, Malik OP, Nouraei Pour SM. DTO: Donkey Theorem Optimization. In: 2019 27th Iranian Conference on Electrical Engineering (ICEE); 2019. p. 1855–1859.

[203] Kusuma PD, Hasibuan FC. Enriched Coati Osprey Algorithm: A Swarm-based Metaheuristic and Its Sensitivity Evaluation of Its Strategy. IAENG International Journal of Applied Mathematics. 2024;54(2).

[204] Zhao W, Wang L, Zhang Z, Fan H, Zhang J, Mirjalili S, et al. Electric eel foraging optimization: A new bio-inspired optimizer for engineering applications. Expert Systems with Applications. 2024;238; p. 122200.

[205] Yilmaz S, Sen S. Electric fish optimization: a new heuristic algorithm inspired by electrolocation. Neural Computing and Applications. 2020;32(15); p. 11543–11578.

[206] Wang G, Deb S, d S Coelho L. Elephant Herding Optimization. In: 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI); 2015. p. 1–5.

[207] Al-Betar MA, Awadallah MA, Braik MS, Makhadmeh S, Doush IA. Elk herd optimizer: a novel nature-inspired metaheuristic algorithm. Artificial Intelligence Review. 2024;57(3); p. 48.

[208] Oyelade ON, Ezugwu AES, Mohamed TIA, Abualigah L. Ebola Optimization Search Algorithm: A New Nature-Inspired Metaheuristic Optimization Algorithm. IEEE Access. 2022;10; p. 16150–16177.

[209] Harifi S, Khalilian M, Mohammadzadeh J, Ebrahimnejad S. Emperor Penguins Colony: a new metaheuristic algorithm for optimization. Evolutionary Intelligence. 2019;12; p. 211–226.

[210] Dhiman G, Kumar V. Emperor penguin optimizer: a bio-inspired algorithm for engineering problems. Knowledge-Based Systems. 2018;159; p. 20–50.

[211] Yang XS, Deb S. Eagle Strategy Using Lévy Walk and Firefly Algorithms for Stochastic Optimization. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer Berlin Heidelberg; 2010. p. 101–111.

[212] Deb S, Fong S, Tian Z. Elephant Search Algorithm for optimization problems. In: 2015 Tenth International Conference on Digital Information Management (ICDIM); 2015. p. 249–255.

[213] Mandal S. Elephant swarm water search algorithm for global optimization. Sādhanā. 2018;43; p. 1–21.

[214] Sur C, Sharma S, Shukla A. Egyptian Vulture Optimization Algorithm – A New Nature Inspired Meta-heuristics for Knapsack Problem. In: The 9th International Conference on Computing and InformationTechnology (IC2IT2013); 2013. p. 227–237.

[215] Cui X, Gao J, Potok TE. A flocking based algorithm for document clustering analysis. Journal of Systems Architecture. 2006;52(8-9); p. 505–515.

[216] Chu Y, Mi H, Liao H, Ji Z, Wu QH. A Fast Bacterial Swarming Algorithm for high-dimensional function optimization. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence); 2008. p. 3135–3140.

[217] Mutazono A, Sugano M, Murata M. Frog call-inspired self-organizing anti-phase synchronization for wireless sensor networks. In: 2009 2nd International Workshop on Nonlinear Dynamics and Synchronization; 2009. p. 81–88.

[218] Azizi M, Talatahari S, Gandomi AH. Fire Hawk Optimizer: A novel metaheuristic algorithm. Artificial Intelligence Review. 2023;56; p. 287–363.

[219] Bellaachia A, Bari A. Flock by Leader: A Novel Machine Learning Biologically Inspired Clustering Algorithm. In: Advances in Swarm Intelligence; 2012. p. 117–126.

[220] Falahah IA, Al-Baik O, Alomari S, Bektemyssova G, Gochhait S, Leonova I, et al. Frilled Lizard Optimization: A Novel Nature-Inspired Metaheuristic Algorithm for Solving Optimization Problems. Preprints. 2024;p. 1–44.

[221] Pan WT. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. Knowledge-Based Systems. 2012;26; p. 69–74.

[222] de Vasconcelos Segundo EH, Mariani VC, dos Santos Coelho L. Design of heat exchangers using Falcon Optimization Algorithm. Applied Thermal Engineering. 2019;156; p. 119–144.

[223] Mohammed H, Rashid T. FOX: a FOX-inspired optimization algorithm. Applied Intelligence. 2023;53; p. 1030–1050.

[224] Li XL, Shao ZJ, Qian JX. Optimizing method based on autonomous animats: Fish-swarm Algorithm. System Engineering Theory and Practice. 2002;22(11); p. 32–38.

[225] Tsai HC, Lin YH. Modification of the fish swarm algorithm with particle swarm optimization formulation and communication behavior. Applied Soft Computing. 2011;11(8); p. 5367 – 5374.

[226] Bastos Filho CJA, de Lima Neto FB, Lins AJCC, Nascimento AIS, Lima MP. A novel search algorithm based on fish school behavior. In: 2008 IEEE International Conference on Systems, Man and Cybernetics; 2008. p. 2646–2651.

[227] Dehghani M, Trojovskỳ P, Malik OP. Green Anaconda Optimization: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems. Biomimetics. 2023;8(1); p. 121.

[228] Alsayyed O, Hamadneh T, Al-Tarawneh H, Alqudah M, Gochhait S, Leonova I, et al. Giant Armadillo Optimization: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems. Biomimetics. 2023;8(8); p. 619.

[229] Min H, Wang Z. Design and analysis of Group Escape Behavior for distributed autonomous mobile robots. In: 2011 IEEE International Conference on Robotics and Automation; 2011. p. 6128–6135.

[230] Mohammadi-Balani A, Dehghan Nayeri M, Azar A, Taghizadeh-Yazdi M. Golden eagle optimizer: A nature-inspired metaheuristic algorithm. Computers & Industrial Engineering. 2021;152; p. 107050.

[231] Chopra N, Mohsin Ansari M. Golden jackal optimization: A novel nature-inspired optimizer for engineering applications. Expert Systems with Applications. 2022;198; p. 116924.

[232] Hu G, Guo Y, Wei G, Abualigah L. Genghis Khan shark optimizer: A novel nature-inspired algorithm for engineering optimization. Advanced Engineering Informatics. 2023;58; p. 102210.

[233] Su S, Wang J, Fan W, Yin X. Good Lattice Swarm Algorithm for Constrained Engineering Design Optimization. In: 2007 International Conference on Wireless Communications, Networking and Mobile Computing; 2007. p. 6421–6424.

[234] Agushaka JO, Ezugwu AE, Abualigah L. Gazelle optimization algorithm: a novel nature-inspired metaheuristic optimizer. Neural Computing and Applications. 2023;35; p. 4099–4131.

[235] De SK. The goat search algorithms. Artificial Intelligence Review. 2022;(56); p. 8265–8301.

[236] Zhou Y, Luo Q, Liu J. Glowworm Swarm Optimization For Optimization Dispatching System Of Public Transit Vehicles. Journal of Theoretical and Applied Information Technology. 2013;52; p. 205–210.

[237] He S, Wu QH, Saunders JR. Group Search Optimizer: An Optimization Algorithm Inspired by Animal Searching Behavior. IEEE Transactions on Evolutionary Computation. 2009;13(5); p. 973–990.

[238] Wang J, Wang D. Particle swarm optimization with a leader and followers. Progress in Natural Science. 2008;18(11); p. 1437–1443.

[239] Abdollahzadeh B, Soleimanian Gharehchopogh F, Mirjalili S. Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. International Journal of Intelligent Systems. 2021;36(10); p. 5887–5958.

[240] Mirjalili S, Mirjalili SM, Lewis A. Grey Wolf Optimizer. Advances in Engineering Software. 2014;69; p. 46–61.

[241] Morais RG, Nedjah N, Mourelle LM. A novel metaheuristic inspired by Hitchcock birds' behavior for efficient optimization of large search spaces of high dimensionality. Soft Computing. 2020;24(8); p. 5633–5655.

[242] Bozorg-Haddad O, Afshar A, Mariño M. Honey-Bees Mating Optimization (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimization. Water Resources Management. 2006;20; p. 661–680.

[243] Yang Y, Chen H, Heidari AA, Gandomi AH. Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. Expert Systems with Applications. 2021;177; p. 114864.

[244] Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H. Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems. 2019;97; p. 849–872.

[245] El-Dosuky M, El-Bassiouny A, Hamza T, Rashad M. New Hoopoe Heuristic Optimization. International Journal of Science and Advanced Technology. 2012;2(9); p. 85–90.

[246] Peraza-Vázquez H, Peña-Delgado A, Merino-Treviño M, Morales-Cepeda AB, Sinha N. A novel metaheuristic inspired by horned lizard defense tactics. Artificial Intelligence Review. 2024;57(3); p. 1–59.

[247] Moldovan D. Horse Optimization Algorithm: A Novel Bio-Inspired Algorithm for Solving Global Optimization Problems. In: Artificial Intelligence and Bioinspired Computational Methods; 2020. p. 195–209.

[248] Oftadeh R, Mahjoob MJ, Shariatpanahi M. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. Computers and Mathematics with Applications. 2010;60(7); p. 2087–2098.

[249] Quijano N, Passino KM. Honey Bee Social Foraging Algorithms for Resource Allocation, Part I: Algorithm and Theory. In: 2007 American Control Conference; 2007. p. 3383–3388.

[250] Chen H, Zhu Y, Hu K, He X. Hierarchical Swarm Model: A New Approach to Optimization. Discrete Dynamics in Nature and Society. 2010;2010; p. 379649.

[251] Ali A, Zafar K, Bakhshi T. On Nature-Inspired Dynamic Route Planning: Hammerhead Shark Optimization Algorithm. In: 2019 15th International Conference on Emerging Technologies (ICET); 2019. p. 1–6.

[252] Anaraki MV, Farzin S. Humboldt Squid Optimization Algorithm (HSOA): A Novel Nature-Inspired Technique for Solving Optimization Problems. IEEE Access. 2023;11; p. 122069–122115.

[253] Maciel O, Valdivia A, Oliva D, Cuevas E, Zaldívar D, Pérez-Cisneros M. A novel hybrid metaheuristic optimization method: hypercube natural aggregation algorithm. Soft Computing. 2020;(24); p. 8823–8856.

[254] Torabi S, Safi-Esfahani F. Improved Raven Roosting Optimization algorithm (IRRO). Swarm and Evolutionary Computation. 2018;40; p. 144–154.

[255] Tang D, Dong S, Jiang Y, Li H, Huang Y. ITGO: Invasive tumor growth optimization algorithm. Applied Soft Computing. 2015;36; p. 670–698.

[256] Chen C, Tsai Y, Liu I, Lai C, Yeh Y, Kuo S, et al. A Novel Metaheuristic: Jaguar Algorithm with Learning Behavior. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics; 2015. p. 1595–1600.

[257] Chou JS, Truong DN. A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. Applied Mathematics and Computation. 2021;389; p. 125535.

[258] Hernández H, Blum C. Distributed graph coloring: an approach based on the calling behavior of Japanese tree frogs. Swarm Intelligence. 2012;6(2); p. 117–150.

[259] Hossein Gandomi A, Hossein Alavi A. Krill herd: A new bio-inspired optimization algorithm. Communications in Nonlinear Science and Numerical Simulation. 2012;17(12); p. 4831–4845.

[260] Dehghani M, Montazeri Z, Bektemyssova G, Malik OP, Dhiman G, Ahmed AEM. Kookaburra Optimization Algorithm: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems. Biomimetics. 2023;8(6); p. 470.

[261] Agbehadji IE, Millham R, Fong S. Kestrel-Based Search Algorithm for Association Rule Mining and Classification of Frequently Changed Items. In: 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN); 2016. p. 356–360.

[262] Biyanto TR, Irawan S, Febrianto HY, Afdanny N, Rahman AH, Gunawan KS, et al. Killer whale algorithm: an algorithm inspired by the life of killer whale. Procedia Computer Science. 2017;124; p. 151–157.

[263] Rajakumar BR. The Lion's Algorithm: A New Nature-Inspired Search Algorithm. Procedia Technology. 2012;6; p. 126–135.

[264] Wang P, Zhu Z, Huang S. Seven-Spot Ladybird Optimization: A Novel and Efficient Metaheuristic Algorithm for Numerical Optimization. The Scientific World Journal. 2013;2013; p. 378515.

[265] Dehghani M, Bektemyssova G, Montazeri Z, Shaikemelev G, Malik OP, Dhiman G. Lyrebird Optimization Algorithm: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems. Biomimetics. 2023;8(6); p. 507.

[266] Hosseini E. Laying chicken algorithm: A new meta-heuristic approach to solve continuous programming problems. Journal of Applied & Computational Mathematics. 2017;6(1); p. 344–351.

[267] Yazdani M, Jolai F. Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm. Journal of Computational Design and Engineering. 2016;3(1); p. 24–36.

[268] Wang B, Jin X, Cheng B. Lion pride optimizer: An optimization algorithm inspired by lion pride behavior. Science China Information Sciences. 2012;55(10); p. 2369–2389.

[269] Chen S. An Analysis of Locust Swarms on Large Scale Global Optimization Problems. In: Artificial Life: Borrowing from Biology; 2009. p. 211–220.

[270] Rabie AH, Mansour NA, Saleh AI. Leopard seal optimization (LSO): A natural inspired meta-heuristic algorithm. Communications in Nonlinear Science and Numerical Simulation. 2023;125; p. 107338.

[271] Cuevas E, Gonzalez A, Zaldivar D, Cisneros M. An optimisation algorithm based on the behaviour of locust swarms. International Journal of Bio-Inspired Computation. 2015;7; p. 402–407.

[272] Zervoudakis K, Tsafarakis S. A mayfly optimization algorithm. Computers & Industrial Engineering. 2020;145; p. 106559.

[273] Mo H, Xu L. Magnetotactic bacteria optimization algorithm for multimodal optimization. In: 2013 IEEE Symposium on Swarm Intelligence (SIS); 2013. p. 240–247.

[274] Wang GG, Deb S, Cui Z. Monarch butterfly optimization. Neural Computing and Applications. 2015;(31); p. 1995–2014.

[275] Duman E, Uysal M, Alkaya AF. Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. Information Sciences. 2012;217; p. 65–77.

[276] Jahani E, Chizari M. Tackling global optimization problems with a novel algorithm – Mouth Brooding Fish algorithm. Applied Soft Computing Journal. 2018;62; p. 987–1002.

[277] Kusuma PD, Kallista M. Migration-Crossover Algorithm: A Swarm-based Metaheuristic Enriched with Crossover Technique and Unbalanced Neighbourhood Search. International Journal of Intelligent Engineering & Systems. 2024;17(1).

[278] Walton S, Hassan O, Morgan K, Brown MR. Modified cuckoo search: A new gradient free optimisation algorithm. Chaos, Solitons and Fractals. 2011;44(9); p. 710–718.

[279] Obagbuwa IC, Adewumi AO. An Improved Cockroach Swarm Optimization. ScientificWorld Journal. 2014;p. 375358.

[280] Mirjalili S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems. 2015;89; p. 228–249.

[281] Alauddin M. Mosquito flying optimization (MFO). In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT); 2016. p. 79–84.

[282] Klein CE, Coelho LDS. Meerkats-inspired algorithm for global optimization problems; 2018. p. 679–684.

[283] Valdez F, Carreon-Ortiz H, Castillo O. Introduction to the Mycorrhiza Optimization Algorithm. In: Mycorrhiza Optimization Algorithm. Springer Nature Switzerland; 2023. p. 1–84.

[284] ul Amir Afsar Minhas F, Arif M. MOX: A novel global optimization algorithm inspired from Oviposition site selection and egg hatching inhibition in mosquitoes. Applied Soft Computing. 2011;11(8); p. 4614–4625.

[285] Faramarzi A, Heidarinejad M, Mirjalili S, Gandomi AH. Marine Predators Algorithm: A nature-inspired metaheuristic. Expert Systems with Applications. 2020;152; p. 113377.

[286] Mucherino A, Seref O. Monkey search: a novel metaheuristic search for global optimization. In: American Institute of Physics. vol. 953; 2007. p. 162–173.

[287] Wang GG. Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Computing. 2018;10(2); p. 151–164.

[288] Abdel-Basset M, Mohamed R, Zidan M, Jameel M, Abouhawwash M. Mantis Search Algorithm: A novel bio-inspired algorithm for global optimization and engineering design problems. Computer Methods in Applied Mechanics and Engineering. 2023;415; p. 116200.

[289] Luo F, Zhao J, Dong ZY. A new metaheuristic algorithm for real-parameter optimization: Natural aggregation algorithm. In: 2016 IEEE Congress on Evolutionary Computation (CEC); 2016. p. 94–103.

[290] Salgotra R, Singh U. The naked mole-rat algorithm. Neural Computing and Applications. 2019;(31); p. 8837—-8857.

[291] Abdel-Basset M, Mohamed R, Jameel M, Abouhawwash M. Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems. Knowledge-Based Systems. 2023;262; p. 110248.

[292] Salih SQ, Alsewari AA. A new algorithm for normal and large-scale optimization problems: Nomadic People Optimizer. Neural Computing and Applications. 2020;(32); p. 10359–10386.

[293] Drias H, Drias Y, Khennak I. A New Swarm Algorithm Based on Orcas Intelligence for Solving Maze Problems. In: Trends and Innovations in Information Systems and Technologies; 2020. p. 788–797.

[294] Maia RD, d Castro LN, Caminhas WM. OptBees - A Bee-Inspired Algorithm for Solving Continuous Optimization Problems. In: 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence; 2013. p. 142–151.

[295] Zhu GY, Zhang WB. Optimal foraging algorithm for global optimization. Applied Soft Computing. 2017;51; p. 294–313.

[296] de Vasconcelos Segundo EH, Mariani VC, dos Santos Coelho L. Metaheuristic inspired on owls behavior applied to heat exchangers design. Thermal Science and Engineering Progress. 2019;14; p. 100431.

[297] Dehghani M, Trojovskỳ P. Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems. Frontiers in Mechanical Engineering. 2023;8; p. 1126450.

[298] Jiang Y, Wu Q, Zhu S, Zhang L. Orca predation algorithm: A novel bio-inspired algorithm for global optimization problems. Expert Systems with Applications. 2022;188; p. 116026.

[299] Kallioras NA, Lagaros ND, Avtzis DN. Pity beetle algorithm – A new metaheuristic inspired by the behavior of bark beetles. Advances in Engineering Software. 2018;121; p. 147–166.

[300] Połap D, Wozniak M. Polar Bear Optimization Algorithm: Meta-Heuristic with Fast Population Movement and Dynamic Birth and Death Mechanism. Symmetry. 2017;9(203); p. 1–20.

[301] Awad A, Coghill GM, Pang W. A novel Physarum-inspired competition algorithm for discrete multi-objective optimisation problems. Soft Computing. 2023;p. 1–21.

[302] Ezugwu AE, Agushaka JO, Abualigah L, Mirjalili S, Gandomi AH. Prairie Dog Optimization Algorithm. Neural Computing and Applications. 2022;34(22); p. 20017–20065.

[303] Duan H, Qiao P. Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. International Journal of Intelligent Computing and Cybernetics. 2014;7(1); p. 24–37.

[304] Zhang W, Luo Q, Zhou Y. A Method for Training RBF Neural Networks Based on Population Migration Algorithm. In: Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence. vol. 1; 2009. p. 165–169.

[305] Abdollahzadeh B, Khodadadi N, Barshandeh S, Trojovskỳ P, Gharehchopogh FS, El-kenawy ESM, et al. Puma optimizer (PO): A novel metaheuristic optimization algorithm and its application in machine learning. Cluster Computing. 2024;p. 1–49.

[306] Trojovskỳ P, Dehghani M. Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications. Sensors. 2022;22(3); p. 855.

[307] Al-Baik O, Alomari S, Alssayed O, Gochhait S, Leonova I, Dutta U, et al. Pufferfish Optimization Algorithm: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems. Biomimetics. 2024;9(2).

[308] Tilahun SL, Choon Ong H. Prey-predator algorithm: A new metaheuristic algorithm for optimization problems. International Journal of Information Technology and Decision Making. 2015;14(6); p. 1331–1352.

[309] Gheraibia Y, Moussaoui A. Penguins Search Optimization Algorithm (PeSOA). In: Recent Trends in Applied Artificial Intelligence; 2013. p. 222–231.

[310] Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization. Soft Computing. 2019;23(3); p. 715–734.

[311] Fard AF, Hajiaghaei-Keshteli M. Red Deer Algorithm (RDA); a new optimization algorithm inspired by Red Deers' mating. In: International Conference on Industrial Engineering, IEEE.,(2016 e); 2016. p. 33–34.

[312] Połap D, Woźniak M. Red fox optimization algorithm. Expert Systems with Applications. 2021;166; p. 114107.

[313] Wang GG, Gao XZ, Zenger K, Coelho LdS. A novel metaheuristic algorithm inspired by rhino herd behavior. In: Proceedings of The 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016, The 57th SIMS Conference on Simulation and Modelling SIMS 2016; 2018. p. 1026–1033.

[314] Khateeb BA, Ahmed K, Mahmood M, Le DN. Rock Hyraxes Swarm Optimization: A New Nature-Inspired Metaheuristic Optimization Algorithm. Computers, Materials & Continua. 2021;68(1); p. 643–654.

[315] Havens T, J Spain C, G Salmon N, M Keller J. Roach Infestation Optimization. In: 2008 IEEE Swarm Intelligence Symposium, SIS 2008; 2008. p. 1–7.

[316] Zangbari Koohi S, Abdul Hamid NAW, Othman M, Ibragimov G. Raccoon Optimization Algorithm. IEEE Access. 2019;7; p. 5383–5399.

[317] Sharma A. A new optimizing algorithm using reincarnation concept. In: 2010 11th International Symposium on Computational Intelligence and Informatics (CINTI); 2010. p. 281–288.

[318] Rabie AH, Saleh AI, Mansour NA. Red piranha optimization (RPO): a natural inspired meta-heuristic algorithm for solving complex optimization problems. Journal of Ambient Intelligence and Humanized Computing. 2023;14(6); p. 7621–7648.

[319] Givi H, Dehghani M, Hubàlovskỳ S. Red Panda Optimization Algorithm: An Effective Bio-Inspired Metaheuristic Algorithm for Solving Engineering Optimization Problems. IEEE Access. 2023;11; p. 57203–57227.

[320] Brabazon A, Cui W, O'neill M. The Raven Roosting Optimisation Algorithm. Soft Computing. 2016;20(2); p. 525–545.

[321] Ferahtia S, Houari A, Rezk H, Djerioui A, Machmoum M, Motahhir S, et al. Red-tailed hawk algorithm for numerical optimization and real-world problems. Scientific Reports. 2023;13(1); p. 12950.

[322] Abualigah L, Elaziz MA, Sumari P, Geem ZW, Gandomi AH. Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. Expert Systems with Applications. 2022;191; p. 116158.

[323] Dhiman G, Garg M, Nagar A, Kumar V, Dehghani M. A novel algorithm for global optimization: rat swarm optimizer. Journal of Ambient Intelligence and Humanized Computing. 2021;12; p. 8457–8482.

[324] Saadi Y, Tri I, Yanto I, Herawan T, Balakrishnan V, Chiroma H, et al. Ringed Seal Search for Global Optimization via a Sensitive Search Model. PLOS ONE. 2015;11(1); p. 1–31.

[325] Hersovici M, Jacovi M, Maarek YS, Pelleg D, Shtalhaim M, Ur S. The shark-search algorithm. An application: tailored Web site mapping. Computer Networks and ISDN Systems. 1998;30(1-7); p. 317–326.

[326] Kusuma PD, Dinimaharawati A. Swarm Bipolar Algorithm: A Metaheuristic Based on Polarization of Two Equal Size Sub Swarms. International Journal of Intelligent Engineering & Systems. 2024;17(2).

[327] McCaffrey JD. Generation of pairwise test sets using a simulated bee colony algorithm. In: 2009 IEEE International Conference on Information Reuse Integration; 2009. p. 115–119.

[328] Samareh Moosavi SH, Khatibi Bardsiri V. Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation. Engineering Applications of Artificial Intelligence. 2017;60; p. 1–15.

[329] Mirjalili S. SCA: A Sine Cosine Algorithm for solving optimization problems. Knowledge-Based Systems. 2016;96; p. 120–133.

[330] Seyyedabbasi A, Kiani F. Sand Cat swarm optimization: A nature-inspired algorithm to solve global optimization problems. Engineering with Computers. 2023;39; p. 2627–2651.

[331] Rakhshani H, Rahati A. Snap-drift cuckoo search: A novel cuckoo search optimization algorithm. Applied Soft Computing. 2017;52; p. 771 – 794.

[332] Eusuff M, Lansey K, Pasha F. Shuffled frog-leaping algorithm: a memetic metaheuristic for discrete optimization. Engineering Optimization. 2006;38(2); p. 129–154.

[333] Dhiman G, Kumar V. Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. Advances in Engineering Software. 2017;114; p. 48–70.

[334] Fausto F, Cuevas E, Valdivia A, González A. A global optimization algorithm inspired in the behavior of selfish herds. Biosystems. 2017;160; p. 39–55.

[335] Zhao S, Zhang T, Ma S, Wang M. Sea-horse optimizer: a novel nature-inspired meta-heuristic for global optimization problems. Applied Intelligence. 2023;53(10); p. 11833–11860.

[336] Su MC, Su SY, Zhao YX. A swarm-inspired projection algorithm. Pattern Recognition. 2009;42(11); p. 2764–2786.

[337] Monismith DR, Mayfield BE. Slime Mold as a model for numerical optimization. In: 2008 IEEE Swarm Intelligence Symposium; 2008. p. 1–8.

[338] Raouf O, M Hezam I. Sperm motility algorithm: a novel metaheuristic approach for global optimisation. International Journal of Operational Research. 2017;28; p. 143.

[339] Chand Bansal J, Sharma H, Singh Jadon S, Clerc M. Spider Monkey Optimization algorithm for numerical optimization. Memetic Computation. 2014;6; p. 31–47.

[340] Zamani H, Nadimi-Shahraki MH, Gandomi AH. Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering optimization. Computer Methods in Applied Mechanics and Engineering. 2022;392; p. 114616.

[341] Dai C, Zhu Y, Chen W. Seeker Optimization Algorithm. In: Computational Intelligence and Security; 2007. p. 167–176.

[342] Dhiman G, Kumar V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. Knowledge-Based Systems. 2019;165; p. 169–196.

[343] Kaur A, Jain S, Goel S. Sandpiper optimization algorithm: a novel approach for solving real-life engineering problems. Applied Intelligence. 2020;50(2); p. 582–619.

[344] Shadravan S, Naji HR, Bardsiri VK. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Engineering Applications of Artificial Intelligence. 2019;80; p. 20–34.

[345] Dehghani M, Trojovskỳ P. Serval Optimization Algorithm: A New Bio-Inspired Approach for Solving Optimization Problems. Biomimetics. 2022;7(4); p. 204.

[346] Cheng MY, Prayogo D. Symbiotic Organisms Search: A new metaheuristic optimization algorithm. Computers and Structures. 2014;139; p. 98–112.

[347] Dhiman G, Kaur A. STOA: A bio-inspired based optimization algorithm for industrial engineering problems. Engineering Applications of Artificial Intelligence. 2019;82; p. 148–174.

[348] Yu JJQ, Li VOK. A social spider algorithm for global optimization. Applied Soft Computing. 2015;30; p. 614–627.

[349] Jain M, Singh V, Rani A. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. Swarm and Evolutionary Computation. 2019;44; p. 148–175.

[350] Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software. 2017;114; p. 163–191.

[351] Xue J, Shen B. A novel swarm intelligence optimization approach: sparrow search algorithm. Systems Science & Control Engineering. 2020;8(1); p. 22–34.

[352] R AL, S P. Sling-shot spider optimization algorithm based packet length control in wireless sensor network and Internet of Things-based networks. International Journal of Communication Systems. 2023;36(4); p. e5406.

[353] Abedinia O, Amjady N, Ghasemi A. A New Metaheuristic Algorithm Based on Shark Smell Optimization. Complexity. 2016;21(5); p. 97–116.

[354] Neshat M, Sepidnam G, Sargolzaei M. Swallow swarm optimization algorithm: a new method to optimization. Neural Computing and Applications. 2013;23(2); p. 429–454.

[355] Cuevas E, Cienfuegos M, Záldivar D, Pérez-Cisneros M. A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Systems with Applications. 2013;40(16); p. 6374–6384.

[356] Shehadeh H, Idris M, Ahmedy I, Ramli R, N M N. The Multi-Objective Optimization Algorithm Based on Sperm Fertilization Procedure (MOSFP) Method for Solving Wireless Sensor Networks Optimization Problems in Smart Grid Applications. Energies. 2018;11; p. 97.

[357] Omidvar R, Parvin H, Rad F. SSPCO Optimization Algorithm (See-See Partridge Chicks Optimization). In: 2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI); 2015. p. 101–106.

[358] Haiyan Q, Xinling S. A Surface-Simplex Swarm Evolution Algorithm. Advances in Engineering Software. 2017;22; p. 38–50.

[359] Trojovskỳ P, Dehghani M, Hanuš P. Siberian Tiger Optimization: A New Bio-Inspired Metaheuristic Algorithm for Solving Engineering Optimization Problems. IEEE Access. 2022;10; p. 132396–132431.

[360] Ebrahimi A, Khamehchi E. Sperm whale algorithm: An effective metaheuristic algorithm for production optimization problems. Journal of Natural Gas Science and Engineering. 2016;29; p. 211–222.

[361] Abdel-Basset M, Mohamed R, Jameel M, Abouhawwash M. Spider wasp optimizer: A novel meta-heuristic optimization algorithm. Artificial Intelligence Review. 2023;56; p. 11675—11738.

[362] Zungeru AM, Ang LM, Seng KP. Termite-hill: Performance optimized swarm intelligence based routing algorithm for wireless sensor networks. Journal of Network and Computer Applications. 2012;35(6); p. 1901–1917.

[363] Majumder A. Termite alate optimization algorithm: a swarm-based nature inspired algorithm for optimization problems. Evolutionary Intelligence. 2023;16(3); p. 997–1017.

[364] Hedayatzadeh R, Akhavan Salmassi F, Keshtgari M, Akbari R, Ziarati K. Termite colony optimization: A novel approach for optimizing continuous problems. In: 2010 18th Iranian Conference on Electrical Engineering; 2010. p. 553–558.

[365] Dehghani M, Hubálovský S, Trojovskỳ P. Tasmanian Devil Optimization: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm. IEEE Access. 2022;10; p. 19599–19620.

[366] Panteleev AV, Kolessa AA. Application of the Tomtit Flock Metaheuristic Optimization Algorithm to the Optimal Discrete Time Deterministic Dynamical Control Problem. Algorithms. 2022;15(9); p. 301.

[367] Mozaffari A, Goudarzi AM, Fathi A. Bio-inspired methods for fast and robust arrangement of thermoelectric modulus. International Journal of Bio-Inspired Computation (IJBIC). 2013;5(1); p. 19–34.

[368] Minh HL, Sang-To T, Theraulaz G, Abdel Wahab M, Cuong-Le T. Termite life cycle optimizer. Expert Systems with Applications. 2023;213; p. 119211.

[369] Sahu VSDM, Samal P, Panigrahi CK. Tyrannosaurus optimization algorithm: A new nature-inspired meta-heuristic algorithm for solving optimal control problems. e-Prime - Advances in Electrical Engineering, Electronics and Energy. 2023;5; p. 100243.

[370] Kaur S, Awasthi LK, Sangal AL, Dhiman G. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Engineering Applications of Artificial Intelligence. 2020;90; p. 103541.

[371] Layeb A. Tangent search algorithm for solving optimization problems. Neural Computing and Applications. 2022;34(11); p. 8853–8884.

[372] Yang X, Lees JM, Morley CT. Application of Virtual Ant Algorithms in the Optimization of CFRP Shear Strengthened Precracked Structures. In: Computational Science - ICCS 2006, 6th International Conference, Proceedings, Part I; 2006. p. 834–837.

[373] Yang XS. Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. In: Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach; 2005. p. 317–323.

[374] Li MD, Zhao H, Weng XW, Han T. A novel nature-inspired algorithm for optimization: Virus colony search. Advances in Engineering Software. 2016;92; p. 65–88.

[375] Juarez JRC, Wang HJ, Lai YC, Liang YC. Virus Optimization Algorithm (VOA): A novel metaheuristic for solving continuous optimization problems. In: Proceedings of the 2009 Asia Pacific Industrial Engineering and Management Systems Conference (APIEMS 2009); 2009. p. 2166–2174.

[376] Cortés P, García JM, Muñuzuri J, Onieva L. Viral systems: A new bio-inspired optimisation approach. Computers and Operations Research. 2008;35; p. 2840–2860.

[377] Theraulaz G, Goss S, Gervet J, Deneubourg JL. Task differentiation in Polistes wasp colonies: a model for self-organizing groups of robots. In: Proceedings of the First International Conference on Simulation of Adaptive Behavior: From Animals to Animates; 1991. p. 346–355.

[378] Liu CY, Yan XH, Wu H. The Wolf Colony Algorithm and Its Application. Chinese Journal of Electronics. 2011;20; p. 212–216.

[379] Arnaout JP. Worm Optimization: A novel optimization algorithm inspired by C. Elegans. In: Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management; 2014. p. 2499–2505.

[380] Mirjalili S, Lewis A. The Whale Optimization Algorithm. Advances in Engineering Software. 2016;95; p. 51–67.

[381] Trojovskỳ P, Dehghani M. A new bio-inspired metaheuristic algorithm for solving optimization problems based on walruses behavior. Scientific Reports. 2023;13; p. 8775.

[382] Yang C, Tu X, Chen J. Algorithm of Marriage in Honey Bees Optimization Based on the Wolf Pack Search. In: Proceedings of the The 2007 International Conference on Intelligent Pervasive Computing; 2007. p. 462–467.

[383] Ting TO, Man KL, Guan SU, Nayel M, Wan K. Weightless Swarm Algorithm (WSA) for Dynamic Optimization Problems. In: Network and Parallel Computing, IFIP International Conference on Network and Parallel Computing; 2012. p. 508–515.

[384] Tang R, Fong S, Yang XS, Deb S. Wolf search algorithm with ephemeral memory. In: Seventh International Conference on Digital Information Management (ICDIM 2012); 2012. p. 165–172.

[385] Pinto P, Runkler TA, Sousa JM. Wasp swarm optimization of logistic systems. In: Adaptive and Natural Computing Algorithms; 2005. p. 264–267.

[386] Braik M, Hammouri A, Atwan J, Al-Betar MA, Awadallah MA. White Shark Optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. Knowledge-Based Systems. 2022;243; p. 108457.

[387] Zaldívar D, Morales B, Rodríguez A, Valdivia-G A, Cuevas E, Pérez-Cisneros M. A novel bio-inspired optimization model based on Yellow Saddle Goatfish behavior. Biosystems. 2018;174; p. 1–21.

[388] Trojovská E, Dehghani M, Trojovskỳ P. Zebra Optimization Algorithm: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm. IEEE Access. 2022;10; p. 49445–49473.

[389] Nguyen HT, Bhanu B. Zombie Survival Optimization: A Swarm Intelligence Algorithm Inspired By Zombie Foraging. In: 21st International Conference on Pattern Recognition (ICPR 2012); 2012. p. 987–990.

[390] Yang XS, Deb S, Mishra SK. Multi-species Cuckoo Search Algorithm for Global Optimization. Cognitive Computation. 2018;10(6); p. 1085–1095.

[391] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: A Gravitational Search Algorithm. Information Sciences. 2009;179(13); p. 2232–2248.

[392] Hatamlou A. Black hole: A new heuristic optimization approach for data clustering. Information Sciences. 2013;222; p. 175–184.

[393] Shah-Hosseini H. Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. International Journal of Computational Science and Engineering. 2011;6(1-2); p. 132–140.

[394] Lee KS, Geem ZW. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering. 2005;194; p. 3902–3933.

[395] Yadav A, Yadav A. AEFA: Artificial electric field algorithm for global optimization. Swarm and Evolutionary Computation. 2019;48; p. 93–108.

[396] Hashim FA, Hussain K, Houssein EH, Mabrouk MS, Al-Atabany W. Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Applied Intelligence. 2021;51; p. 1531–1551.

[397] Xie L, Zeng J, Cui Z. General framework of Artificial Physics Optimization Algorithm. In: 2009 World Congress on Nature Biologically Inspired Computing (NaBIC); 2009. p. 1321–1326.

[398] Zhao W, Wang L, Zhang Z. Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. Knowledge-Based Systems. 2019;163; p. 283–304.

[399] Erol OK, Eksin I. A new optimization method: Big Bang–Big Crunch. Advances in Engineering Software. 2006;37(2); p. 106–111.

[400] Kaveh A, Mahdavi VR. Colliding bodies optimization: A novel meta-heuristic method. Computers and Structures. 2014;139; p. 18–27.

[401] Feng X, Ma M, Yu H. Crystal Energy Optimization Algorithm. Computational Intelligence. 2016;32(2); p. 284–322.

[402] Formato RA. Central Force Optimization: A New Nature Inspired Computational Framework for Multidimensional Search and Optimization. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2007). Springer Berlin Heidelberg; 2008. p. 221–238.

[403] Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search. Acta Mechanica. 2010;213(3-4); p. 267–289.

[404] Abedinpourshotorban H, Shamsuddin SM, Beheshti Z, Jawawi DNA. Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm. Swarm and Evolutionary Computation. 2016;26; p. 8–22.

[405] Ilker Birbil S, Fang SC. An Electromagnetism-like Mechanism for Global Optimization. Journal of Global Optimization. 2003;25(3); p. 263–282.

[406] Khalafallah A, Abdel-Raheem M. Electimize: new evolutionary algorithm for optimization with application in construction engineering. Journal of Computing in Civil Engineering. 2011;25(3); p. 192–201.

[407] Rahmanzadeh S, Pishvaee MS. Electron radar search algorithm: a novel developed meta-heuristic algorithm. Soft Computing. 2020;24(11); p. 8443–8465.

[408] Kundu S. Gravitational clustering: A new approach based on the spatial distribution of the points. Pattern Recognition. 1999;32(7); p. 1149–1160.

[409] Barzegar B, Rahmani AM, Far KZ. Gravitational emulation local search algorithm for advanced reservation and scheduling in grid systems. In: 2009 First Asian Himalayas International Conference on Internet; 2009. p. 1–5.

[410] Zheng M, Liu Gx, Zhou Cg, Liang Yc, Wang Y. Gravitation field algorithm and its application in gene cluster. Algorithms for Molecular Biology. 2010;5(32); p. 1–11.

[411] Ghasemi M, Zare M, Zahedi A, Akbari MA, Mirjalili S, Abualigah L. Geyser inspired algorithm: A new geological-inspired meta-heuristic for real-parameter and constrained engineering optimization. Journal of Bionic Engineering. 2023;p. 1–35.

[412] Flores JJ, López R, Barrera J. Gravitational Interactions Optimization. In: Learning and Intelligent Optimization; 2011. p. 226–237.

[413] Beiranvand H, Rokrok E. General Relativity Search Algorithm: A Global Optimization Approach. International Journal of Computational Intelligence and Applications. 2015;14(3); p. 1–29.

[414] Muthiah-Nakarajan V, Noel MM. Galactic Swarm Optimization: A new global optimization metaheuristic inspired by galactic motion. Applied Soft Computing. 2016;38; p. 771–787.

[415] Wedyan A, Whalley J, Narayanan A. Hydrological Cycle Algorithm for Continuous Optimization Problems. Journal of Optimization. 2017;2017; p. 1–25.

[416] Cui Y, Guo R, Guo D. Lambda algorithm. Journal of Uncertain Systems. 2010;4(1); p. 22–33.

[417] Zaránd G, Pázmándi F, Pál KF, Zimányi GT. Using Hysteresis for Optimization. Physical Review Letters. 2002;89(15); p. 150201.

[418] Rbouh I, El Imrani AA. Hurricane-based Optimization Algorithm. AASRI Procedia. 2014;6; p. 26–33.

[419] Askari H, Zahiri SH. Intelligent Gravitational Search Algorithm for optimum design of fuzzy classifier. In: 2012 2nd International eConference on Computer and Knowledge Engineering, ICCKE 2012; 2012. p. 98–104.

[420] Shah-Hosseini H. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. International Journal of Bio-inspired computation. 2009;1(1); p. 71–79.

[421] Jihong Shen, Jialian Li. The principle analysis of Light Ray Optimization Algorithm. In: 2010 Second International Conference on Computational Intelligence and Natural Computing. vol. 2; 2010. p. 154–157.

[422] Shareef H, Ibrahim AA, Mutlag AH. Lightning search algorithm. Applied Soft Computing. 2015;36; p. 315–333.

[423] Tayarani-N MH, Akbarzadeh-T MR. Magnetic Optimization Algorithms a new synthesis. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence); 2008. p. 2659–2664.

[424] Mora-Gutiérrez RA, Ramírez-Rodríguez J, Rincón-García EA. An optimization algorithm inspired by musical composition. Artificial Intelligence Review. 2014;41(3); p. 301–315.

[425] Ashrafi SM, Dariane AB. A novel and effective algorithm for numerical optimization: Melody Search (MS). In: 2011 11th International Conference on Hybrid Intelligent Systems (HIS); 2011. p. 109–114.

[426] Mirjalili S, Mirjalili SM, Hatamlou A. Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. Neural Computing and Applications. 2016;27(2); p. 495–513.

[427] Sowmya R, Premkumar M, Jangir P. Newton-Raphson-based optimizer: A new population-based metaheuristic algorithm for continuous optimization problems. Engineering Applications of Artificial Intelligence. 2024;128; p. 107532.

[428] Kashan AH. A new metaheuristic for optimization: Optics inspired optimization (OIO). Computers and Operations Research. 2015;55; p. 99–125.

[429] Sacco WF, Filho HA, De Oliveira CRE. A populational particle collision algorithm applied to a nuclear reactor core design optimization. In: Joint International Topical Meeting on Mathematics and Computations and Supercomputing in Nuclear Applications, 2007; 2007. p. 1–10.

[430] Taillard ÉD, Voss S. Popmusic — Partial Optimization Metaheuristic under Special Intensification Conditions. In: Essays and Surveys in Metaheuristics. Springer US; 2002. p. 613–629.

[431] Saire JEC, Túpac VYJ. An approach to real-coded quantum inspired evolutionary algorithm using particles filter. In: 2015 Latin America Congress on Computational Intelligence (LA-CCI); 2015. p. 1–6.

[432] Kaboli SHA, Selvaraj J, Rahim NA. Rain-fall optimization algorithm: A population based algorithm for solving constrained optimization problems. Journal of Computational Science. 2017;19; p. 31–42.

[433] Biyanto TR, Matradji, Febrianto HY, Afdanny N, Rahman AH, Gunawan KS. Rain Water Algorithm: Newton's Law of Rain Water Movements during Free Fall and Uniformly Accelerated Motion Utilization. AIP Conference Proceedings. 2019;2088(1); p. 020053.

[434] Rabanal P, Rodríguez I, Rubio F. Using River Formation Dynamics to Design Heuristic Algorithms. In: Unconventional Computation; 2007. p. 163–177.

[435] Rahmani R, Yusof R. A new simple, fast and efficient algorithm for global optimization over continuous search-space problems: Radial Movement Optimization. Applied Mathematics and Computation. 2014;248; p. 287–300.

[436] Kaveh A, Khayatazad M. A new meta-heuristic method: Ray Optimization. Computers and Structures. 2012;112–113; p. 283–294.

[437] Deng L, Liu S. Snow ablation optimizer: A novel metaheuristic technique for numerical optimization and engineering design. Expert Systems with Applications. 2023;225; p. 120069.

[438] Hsiao YT, Chuang CL, Jiang JA, Chien CC. A novel optimization algorithm: space gravitational optimization. In: 2005 IEEE International Conference on Systems, Man and Cybernetics. vol. 3; 2005. p. 2323–2328.

[439] Tzanetos A, Dounias G. A new metaheuristic method for optimization: sonar inspired optimization. In: International Conference on Engineering Applications of Neural Networks; 2017. p. 417–428.

[440] Cuevas E, Echavarría A, Ramírez-Ortegón MA. An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation. Applied Intelligence. 2014;40; p. 256–272.

[441] Tamura K, Yasuda K. Primary Study of Spiral Dynamics Inspired Optimization. IEEJ Transactions On Electrical And Electronic Engineering. 2011;6; p. 98–100.

[442] Jin GG, Tran TD. A nature-inspired evolutionary algorithm based on spiral movements. In: Proceedings of SICE Annual Conference 2010; 2010. p. 1643–1647.

[443] Vicsek T, Czirók A, Ben-Jacob E, Cohen I, Shochet O. Novel Type of Phase Transition in a System of Self-Driven Particles. Physical Review Letters. 1995;75(6); p. 1226–1229.

[444] Zitouni F, Harous S, Maamri R. The Solar System Algorithm: A Novel Metaheuristic Method for Global Optimization. IEEE Access. 2021;9; p. 4542–4565.

[445] Ghasemi M, Davoudkhani IF, Akbari E, Rahimnejad A, Ghavidel S, Li L. A novel and effective optimization algorithm for global optimization and its engineering applications: Turbulent Flow of Water-based Optimization (TFWO). Engineering Applications of Artificial Intelligence. 2020;92; p. 103666.

[446] Kaveh A, Ilchi Ghazaan M. Vibrating particles system algorithm for truss optimization with multiple natural frequency constraints. Acta Mechanica. 2017;228; p. 307–322.

[447] Dogan B, Ölmez T. A new metaheuristic for numerical function optimization: Vortex Search algorithm. Information Sciences. 2015;293; p. 125–145.

[448] Eskandar H, Sadollah A, Bahreininejad A, Hamdi M. Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. Computers and Structures. 2012;110–111; p. 151–166.

[449] Kaveh A, Bakhshpoori T. Water Evaporation Optimization: A novel physically inspired optimization algorithm. Computers and Structures. 2016;167; p. 69–85.

[450] Yang FC, Wang YP. Water flow-like algorithm for object grouping problems. Journal of the Chinese Institute of Industrial Engineers. 2007;24(6); p. 475–488.

[451] Basu S, Chaudhuri C, Kundu M, Nasipuri M, Basu DK. Text line extraction from multi-skewed handwritten documents. Pattern Recognition. 2007;40(6); p. 1825–1839.

[452] Tran TH, Ng KM. A water-flow algorithm for flexible flow shop scheduling with intermediate buffers. Journal of Scheduling. 2011;14(5); p. 483–500.

[453] Zheng YJ. Water wave optimization: A new nature-inspired metaheuristic. Computers and Operations Research. 2015;55; p. 1–11.

[454] Irizarry R. A generalized framework for solving dynamic optimization problems using the artificial chemical process paradigm: Applications to particulate processes and discrete dynamic systems. Chemical Engineering Science. 2005;60(21); p. 5663–5681.

[455] Alatas B. ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization. Expert Systems with Applications. 2011;38; p. 13170–13180.

[456] Melin P, Astudillo L, Castillo O, Valdez F, Garcia M. Optimal design of type-2 and type-1 fuzzy tracking controllers for autonomous mobile robots under perturbed torques using a new chemical optimization paradigm. Expert Systems with Applications. 2013;40(8); p. 3185–3195.

[457] Lam AYS, Li VOK. Chemical-Reaction-Inspired Metaheuristic for Optimization. IEEE Transactions on Evolutionary Computation. 2010;14(3); p. 381–399.

[458] Abdechiri M, Meybodi MR, Bahrami H. Gases Brownian Motion Optimization: an Algorithm for Optimization (GBMO). Applied Soft Computing. 2013;13; p. 2932–2946.

[459] Patel VK, Savsani VJ. Heat transfer search (HTS): a novel optimization algorithm. Information Science. 2015;324; p. 217–246.

[460] Javidy B, Hatamlou A, Mirjalili S. Ions motion algorithm for solving optimization problems. Applied Soft Computing. 2015;32; p. 72–79.

[461] Chuang CL, Jiang JA. Integrated radiation optimization: inspired by the gravitational radiation in the curvature of space-time. In: 2007 IEEE Congress on Evolutionary Computation; 2007. p. 3157–3164.

[462] Moein S, Logeswaran R. KGMO: A swarm optimization algorithm based on the kinetic energy of gas molecules. Information Sciences. 2014;275; p. 127–144.

[463] Murase H. Finite element inverse analysis using a photosynthetic algorithm. Computers and Electronics in Agriculture. 2000;29(1-2); p. 115–123.

[464] Subashini P, Dhivyaprabha TT, Krishnaveni M. Synergistic Fibroblast Optimization. In: Artificial Intelligence and Evolutionary Computations in Engineering Systems; 2017. p. 285–294.

[465] Kaveh A, Dadras A. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. Advances in Engineering Software. 2017;110; p. 69–84.

[466] Huan TT, Kulkarni AJ, Kanesan J, Huang CJ, Abraham A. Ideology algorithm: a socio-inspired optimization methodology. Neural Computing and Applications. 2017;28(1); p. 845–876.

[467] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In: 2007 IEEE Congress on Evolutionary Computation; 2007. p. 4661–4667.

[468] Moosavian N, Roodsari BK. Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. Swarm and Evolutionary Computation. 2014;17; p. 14–24.

[469] Shi Y. Brain Storm Optimization Algorithm. In: Advances in Swarm Intelligence; 2011. p. 303–309.

[470] El-Abd M. Global-best brain storm optimization algorithm. Swarm and Evolutionary Computation. 2017;37; p. 27 – 44.

[471] Bogar E, Beyhan S. Adolescent Identity Search Algorithm (AISA): A novel metaheuristic approach for solving optimization problems. Applied Soft Computing. 2020;95; p. 106503.

[472] Ahmadi-Javid A. Anarchic Society Optimization: A human-inspired method. In: 2011 IEEE Congress of Evolutionary Computation (CEC); 2011. p. 2586–2592.

[473] Yuan Y, Ren J, Wang S, Wang Z, Mu X, Zhao W. Alpine skiing optimization: A new bio-inspired optimization algorithm. Advances in Engineering Software. 2022;170; p. 103158.

[474] Bodaghi M, Samieefar K. Meta-heuristic bus transportation algorithm. Iran Journal of Computer Science. 2019;2; p. 23–32.

[475] Zhang Q, Wang R, Yang J, Ding K, Li Y, Hu J. Collective decision optimization algorithm: A new heuristic optimization method. Neurocomputing. 2017;221; p. 123–137.

[476] Li M, Zhao H, Weng X, Han T. Cognitive behavior optimization algorithm for solving optimization problems. Applied Soft Computing. 2016;39; p. 199–222.

[477] Sharafi Y, Khanesar MA, Teshnehlab M. COOA: Competitive optimization algorithm. Swarm and Evolutionary Computation. 2016;30; p. 39–63.

[478] Milani A, Santucci V. Community of scientist optimization: An autonomy oriented approach to distributed optimization. AI Communications. 2012;25; p. 157–172.

[479] Xidong Jin, Reynolds RG. Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99. vol. 3; 1999. p. 1672–1678.

[480] Biyanto TR, Fibrianto HY, Nugroho G, Hatta AM, Listijorini E, Budiati T, et al. Duelist algorithm: an algorithm inspired by how duelist improve their capabilities in a duel. In: International Conference on Swarm Intelligence; 2016. p. 39–47.

[481] Emami H, Derakhshan F. Election algorithm: A new socio-politically inspired strategy. AI Communications. 2015;28(3); p. 591–603.

[482] Fadakar E, Ebrahimi M. A new metaheuristic football game inspired algorithm. In: 2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC); 2016. p. 6–11.

[483] Razmjooy N, Khalilpour M, Ramezan M. A New Meta-Heuristic Optimization Algorithm Inspired by FIFA World Cup Competitions: Theory and Its Application in PID Designing for AVR System. Journal of Control, Automation and Electrical Systems. 2016;27(4); p. 419–440.

[484] Osaba E, Diaz F, Onieva E. Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts. Applied Intelligence. 2014;41(1); p. 145–166.

[485] Eita MA, Fahmy MM. Group counseling optimization. Applied Soft Computing. 2010;22; p. 585–604.

[486] Daskin A, Kais S. Group leaders optimization algorithm. Molecular Physics. 2011;109(5); p. 761–772.

[487] Lenord Melvix JSM. Greedy Politics Optimization: Metaheuristic inspired by political strategies adopted during state assembly elections. In: 2014 IEEE International Advance Computing Conference (IACC); 2014. p. 1157–1162.

[488] Kapoor M, Pathak BK, Kumar R. A nature-inspired meta-heuristic knowledge-based algorithm for solving multiobjective optimization problems. Journal of Engineering Mathematics. 2023;143; p. 5.

[489] Zhang Y, Jin Z. Group Teaching Optimization Algorithm: A Novel Metaheuristic Method for Solving Global Optimization Problems. Expert Systems with Applications. 2020;148; p. 113246.

[490] Montiel O, Castillo O, Melin P, Rodríguez Díaz A, Sepúlveda R. Human evolutionary model: A new approach to optimization. Information Sciences. 2007;177(10); p. 2075–2098.

[491] Thammano A, Moolwong J. A new computational intelligence technique based on human group formation. Expert Systems with Applications. 2010;37(2); p. 1628–1634.

[492] Zhang LM, Dahlmann C, Zhang Y. Human-Inspired Algorithms for continuous function optimization. In: 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems; 2009. p. 318–321.

[493] Ghasemian H, Ghasemian F, Vahdat-Nejad H. Human urbanization algorithm: A novel metaheuristic approach. Mathematics and Computers in Simulation. 2020;178; p. 1–15.

[494] Srivastava A, Das DK. A new Kho-Kho optimization Algorithm: An application to solve combined emission economic dispatch and combined heat and power economic dispatch problem. Engineering Applications of Artificial Intelligence. 2020;94; p. 103763.

[495] Kashan AH. League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships. Applied Soft Computing. 2014;16; p. 171–200.

[496] Khatri A, Gaba A, Rana K, Kumar V. A novel life choice-based optimizer. Soft Computing. 2020;24(12); p. 9121–9141.

[497] Gonzalez-Fernandez Y, Chen S. Leaders and followers - A new metaheuristic to avoid the bias of accumulated information. In: 2015 IEEE Congress on Evolutionary Computation (CEC); 2015. p. 776–783.

[498] Hu TC, Kahng AB, Tsao CWA. Old Bachelor Acceptance: A New Class of Non-Monotone Threshold Accepting Methods. ORSA Journal on Computing. 1995;7(4); p. 417–425.

[499] Zhang X, Chen W, Dai C. Application of oriented search algorithm in reactive power optimization of power system. In: 2008 Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies; 2008. p. 2856–2861.

[500] Askari Q, Younas I, Saeed M. Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. Knowledge-Based Systems. 2020;195; p. 105709.

[501] Borji A, Hamide M. A new approach to global optimization motivated by parliamentary political competitions. International Journal of Innovative Computing, Information and Control. 2009;5; p. 1643–1653.

[502] Samareh Moosavi SH, Bardsiri VK. Poor and rich optimization algorithm: A new human-based and multi populations algorithm. Engineering Applications of Artificial Intelligence. 2019;86; p. 165–181.

[503] Zhang J, Xiao M, Gao L, Pan Q. Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems. Applied Mathematical Modelling. 2018;63; p. 464–490.

[504] Shabani A, Asgarian B, Gharebaghi SA, Salido MA, Giret A. A new optimization algorithm based on search and rescue operations. Mathematical Problems in Engineering. 2019;2019; p. 1–24.

[505] Ray T, Liew KM. Society and Civilization: An Optimization Algorithm Based on the Simulation of Social Behavior. IEEE Transactions On Evolutionary Computation. 2003;7(4); p. 386–396.

[506] Xie XF, Zhang WJ, Yang ZL. Social cognitive optimization for nonlinear programming problems. In: Proceedings. International Conference on Machine Learning and Cybernetics. vol. 2; 2002. p. 779—783.

[507] Wei Z, Cui Z, Zeng J. Social Cognitive Optimization Algorithm with Reactive Power Optimization of Power System. In: 2010 International Conference on Computational Aspects of Social Networks; 2010. p. 11–14.

[508] Xu Y, Cui Z, Zeng J. Social Emotional Optimization Algorithm for Nonlinear Constrained Optimization Problems. In: Swarm, Evolutionary, and Memetic Computing; 2010. p. 583–590.

[509] Emami H. Stock exchange trading optimization algorithm: a human-inspired method for global optimization. The Journal of Supercomputing. 2022;78(2); p. 2125–2174.

[510] Weibo W, Quanyuan F, Yongkang Z. A novel particle swarm optimization algorithm with stochastic focusing search for real-parameter optimization. In: 2008 11th IEEE Singapore International Conference on Communication Systems; 2008. p. 583–587.

[511] Dwi Purnomo H. Soccer Game Optimization: Fundamental Concept. Jurnal Sistem Komputer. 2012;4(1); p. 25–36.

[512] Das B, Mukherjee V, Das D. Student psychology based optimization algorithm: A new population based optimization algorithm for solving optimization problems. Advances in Engineering Software. 2020;146; p. 102804.

[513] Nemati M, Zandi Y, Agdas AS. Application of a novel metaheuristic algorithm inspired by stadium spectators in global optimization problems. Scientific Reports. 2024;14; p. 3078.

[514] Rashid MFFA. Tiki-taka algorithm: a novel metaheuristic inspired by football playing style. Engineering Computations. 2020;38; p. 313–343.

[515] Mahmoodabadi MJ, Rasekh M, Zohari T. TGA: Team game algorithm. Future Computing and Informatics Journal. 2018;3(2); p. 191–199.

[516] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer-Aided Design. 2011;43(3); p. 303–315.

[517] Bagheri H, Ara AL, Hosseini R. Thieves and Police, a New Optimization Algorithm: Theory and Application in Probabilistic Power Flow. IETE Journal of Research. 2021;67(6); p. 951–968.

[518] Li Z, Gao X, Huang X, Gao J, Yang X, Li MJ. Tactical unit algorithm: A novel metaheuristic algorithm for optimal loading distribution of chillers in energy optimization. Applied Thermal Engineering. 2024;238; p. 122037.

[519] Kaveh A, Zolghadr A. A Novel Meta-Heuristic Algorithm: Tug Of War Optimization. International Journal Of Optimization In Civil Engineering. 2014;6(4); p. 469–492.

[520] Ardjmand E, Amin-Naseri MR. Unconscious Search - A New Structured Search Algorithm for Solving Continuous Engineering Optimization Problems Based on the Theory of Psychoanalysis. In: Advances in Swarm Intelligence; 2012. p. 233–242.

[521] Moghdani R, Salimifard K. Volleyball Premier League Algorithm. Applied Soft Computing. 2018;64; p. 161–185.

[522] Yampolskiy RV, EL-Barkouky A. Wisdom of artificial crowds algorithm for solving NP-hard problems. International Journal of Bio-Inspired Computation. 2011;3(6); p. 358–369.

[523] Ghaemi M, Feizi-Derakhshi MR. Forest Optimization Algorithm. Expert Systems with Applications. 2014;41(15); p. 6676–6687.

[524] Cheng L, Wu Xh, Wang Y. Artificial Flora (AF) Optimization Algorithm. Applied Sciences. 2018;8(3); p. 329.

[525] Zhao Z, Cui Z, Zeng J, Yue X. Artificial Plant Optimization Algorithm for Constrained Optimization Problems. In: 2011 Second International Conference on Innovations in Bio-inspired Computing and Applications; 2011. p. 120–123.

[526] Ghaemidizaji M, Dadkhah C, Leung H. A new optimization algorithm based on the behavior of BrunsVigia flower. In: 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC); 2018. p. 263–267.

[527] Ong KM, Ong P, Sia CK. A carnivorous plant algorithm for solving global optimization problems. Applied Soft Computing. 2021;98; p. 106833.

[528] Korani W, Mouhoub M. Discrete Mother Tree Optimization for the Traveling Salesman Problem. In: Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 23–27, 2020, Proceedings, Part II; 2020. p. 25—-37.

[529] Yang XS. Flower Pollination Algorithm for Global Optimization. In: Unconventional Computation and Natural Computation, Proceeding; 2012. p. 240–249.

[530] Dalirinia E, Jalali M, Yaghoobi M, Tabatabaee H. Lotus effect optimization algorithm (LEA): a lotus nature-inspired algorithm for engineering design optimization. The Journal of Supercomputing. 2023;80; p. 761–799.

[531] Moez H, Kaveh A, Taghizadieh N. Natural Forest Regeneration Algorithm: A New Meta-Heuristic. Iranian Journal of Science and Technology, Transactions of Civil Engineering. 2016;40(4); p. 311–326.

[532] Cai W, Yang W, Chen X. A Global Optimization Algorithm Based on Plant Growth Theory: Plant Growth Optimization. In: 2008 International Conference on Intelligent Computation Technology and Automation (ICICTA). vol. 1; 2008. p. 1194–1199.

[533] Sulaiman M, Salhi A, Selamoglu BI, Kirikchi OB. A Plant Propagation Algorithm for Constrained Engineering Optimisation Problems. Mathematical Problems in Engineering. 2014;2014; p. 1–10.

[534] Premaratne U, Samarabandu J, Sidhu T. A new biologically inspired optimization algorithm. In: 2009 International Conference on Industrial and Information Systems (ICIIS); 2009. p. 279–284.

[535] Xiaoxian He, Shigeng Zhang, Jie Wang. A novel algorithm inspired by plant root growth with self-similarity propagation. In: 2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom); 2015. p. 157–162.

[536] Labbi Y, ben attous D, A Gabbar H, Mahdad B, Zidan A. A new rooted tree optimization algorithm for economic dispatch with valve-point effect. International Journal of Electrical Power & Energy Systems. 2016;79; p. 298–311.

[537] Merrikh-Bayat F. The runner-root algorithm: A metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature. Applied Soft Computing. 2015;33; p. 292–303.

[538] Karci A. Theory of Saplings Growing Up Algorithm. In: Adaptive and Natural Computing Algorithms; 2007. p. 450–460.

[539] Caraveo C, Valdez F, Castillo O. A new optimization meta-heuristic algorithm based on self-defense mechanism of the plants with three reproduction operators. Soft Computing. 2018;22(15); p. 4907–4920.

[540] Emami H. Seasons optimization algorithm. Engineering with Computers. 2022;38(2); p. 1845–1865.

[541] Merrikh-Bayat F. A Numerical Optimization Algorithm Inspired by the Strawberry Plant; 2014.

[542] Naseri NK, Sundararajan EA, Ayob M, Jula A. Smart Root Search (SRS): A Novel Nature-Inspired Search Algorithm. Symmetry. 2020;12(12); p. 2025.

[543] Cheraghalipour A, Hajiaghaei-Keshteli M, Paydar MM. Tree Growth Algorithm (TGA): A novel approach for solving optimization problems. Engineering Applications of Artificial Intelligence. 2018;72; p. 393–414.

[544] Halim H, Ismail I. Tree Physiology Optimization in Constrained Optimization Problem. Telkomnika (Telecommunication Computing Electronics and Control). 2018;16; p. 876–882.

[545] Kiran MS. TSA: Tree-seed algorithm for continuous optimization. Expert Systems with Applications. 2015;42(19); p. 6686–6698.

[546] Punnathanam V, Kotecha P. Yin-Yang-pair Optimization: A novel lightweight optimization algorithm. Engineering Applications of Artificial Intelligence. 2016;54; p. 62–79.

[547] Felipe D, Goldbarg EFG, Goldbarg MC. Scientific algorithms for the Car Renter Salesman Problem. In: 2014 IEEE Congress on Evolutionary Computation (CEC); 2014. p. 873–879.

[548] Fathollahi-Fard AM, Hajiaghaei-Keshteli M, Tavakkoli-Moghaddam R. The Social Engineering Optimizer (SEO). Engineering Applications of Artificial Intelligence. 2018;72; p. 267–293.

[549] Salimi H. Stochastic Fractal Search: A powerful metaheuristic algorithm. Knowledge-Based Systems. 2015;75; p. 1–18.

[550] Toz M, Toz G. Re-formulated snowflake optimization algorithm (SFO-R). Evolutionary Intelligence. 2023;p. 1–20.

[551] Gonçalves MS, Lopez RH, Fadel Miguel LF. Search group algorithm: A new metaheuristic method for the optimization of truss structures. Computers and Structures. 2015;153; p. 165–184.

[552] Hasançebi O, Azad SK. An efficient metaheuristic algorithm for engineering optimization: SOPT. International Journal of Optimization in Civil Engineering. 2012;2(4); p. 479–487.

[553] Chu SC, Wang TT, Yildiz AR, Pan JS. Ship Rescue Optimization: A New Metaheuristic Algorithm for Solving Engineering Problems. Journal of Internet Technology. 2024;25(1); p. 61–78.

[554] Du H, Wu X, Zhuang J. Small-World Optimization Algorithm for Function Optimization. In: Advances in Natural Computation; 2006. p. 264–273.

[555] Dueck G. New optimization heuristics; The great deluge algorithm and the record-to-record travel. Journal of Computational Physics. 1993;104(1); p. 86–92.

[556] Bayraktar Z, Komurcu M, Werner DH. Wind Driven Optimization (WDO): A novel nature-inspired optimization algorithm and its application to electromagnetics. In: 2010 IEEE Antennas and Propagation Society International Symposium; 2010. p. 1–4.

[557] Gao-Wei Y, Zhanju H. A Novel Atmosphere Clouds Model Optimization Algorithm. In: 2012 International Conference on Computing, Measurement, Control and Sensor Network; 2012. p. 217–220.

[558] Civicioglu P. Artificial cooperative search algorithm for numerical optimization problems. Information Sciences. 2012;229; p. 58–76.

[559] Pijarski P, Kacejko P. A new metaheuristic optimization method: the algorithm of the innovative gunner (AIG). Engineering Optimization. 2019;51(12); p. 1–21.

[560] Wu G. Across neighborhood search for numerical optimization. Information Sciences. 2016;329; p. 597–618.

[561] Hubálovská M, Hubálovskỳ Š, Trojovskỳ P. Botox Optimization Algorithm: A New Human-Based Metaheuristic Algorithm for Solving Optimization Problems. Biomimetics. 2024;9(3); p. 137.

[562] Rahkar Farshi T. Battle royale optimization algorithm. Neural Computing and Applications. 2021;33(4); p. 1139–1157.

[563] Del Acebo E, De La Rosa JL. Introducing bar systems: A class of swarm intelligence optimization algorithms. In: AISB 2008 Convention: Communication, Interaction and Social Intelligence - Proceedings of the AISB 2008 Symposium on Swarm Intelligence Algorithms and Applications; 2008. p. 18–23.

[564] Civicioglu P. Backtracking Search Optimization Algorithm for numerical optimization problems. Applied Mathematics and Computation. 2012;219(15); p. 8121–8144.

[565] Zhu C, Ni J. Cloud Model-Based Differential Evolution Algorithm for Optimization Problems. In: 2012 Sixth International Conference on Internet Computing for Science and Engineering; 2012. p. 55–59.

[566] Li B, Jiang W. Optimizing complex functions by chaos search. Cybernetics and Systems. 1998;29(4); p. 409–419.

[567] Nunes de Castro L, Von Zuben FJ. The Clonal Selection Algorithm with Engineering Applications. In: Workshop Proceedings of GECCO. vol. 10; 2000. p. 36–37.

[568] Hosseini E, Ghafoor KZ, Sadiq AS, Guizani M, Emrouznejad A. COVID-19 Optimizer Algorithm, Modeling and Controlling of Coronavirus Distribution Process. IEEE Journal of Biomedical and Health Informatics. 2020;24(10); p. 2765–2775.

[569] Dehghani M, Montazeri Z, Malik OP. DGO: Dice game optimizer. Gazi University Journal of Science. 2019;32(3); p. 871–882.

[570] Kadioglu S, Sellmann M. Dialectic Search. In: Principles and Practice of Constraint Programming - CP 2009; 2009. p. 486–500.

[571] Civicioglu P. transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. Computers and Geosciences. 2012;46; p. 229–247.

[572] Ghorbani N, Babaei E. Exchange market algorithm. Applied Soft Computing. 2014;19; p. 177–187.

[573] Boettcher S, Percus AG. Extremal Optimization: Methods Derived from Co-evolution. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation. vol. 1; 1999. p. 825–832.

[574] Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S. Equilibrium optimizer: A novel optimization algorithm. Knowledge-Based Systems. 2020;191; p. 105190.

[575] Tan Y, Zhu Y. Fireworks Algorithm for Optimization. In: Advances in Swarm Intelligence; 2010. p. 355–364.

[576] Shayanfar H, Gharehchopogh FS. Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems. Applied Soft Computing. 2018;71; p. 728–746.

[577] Ahrari A, Atai AA. Grenade Explosion Method—A novel tool for optimization of multimodal functions. Applied Soft Computing. 2010;10; p. 1132–1140.

[578] Tanyildizi E, Demir G. Golden sine algorithm: a novel math-inspired algorithm. Advances in Electrical and Computer Engineering. 2017;17(2); p. 71–79.

[579] Husseinzadeh Kashan A, Karimiyan S, Kulkarni AJ. The Golf Sport Inspired Search metaheuristic algorithm and the game theoretic analysis of its operators' effectiveness. Soft Computing. 2024;28(2); p. 1073–1125.

[580] Hatamlou A. Heart: a novel optimization algorithm for cluster analysis. Progress in Artificial Intelligence. 2014;2(2); p. 167–173.

[581] Sellmann M, Tierney K. Hyper-parameterized Dialectic Search for Non-linear Box-Constrained Optimization with Heterogenous Variable Types. In: Learning and Intelligent Optimization; 2020. p. 102–116.

[582] Hosseini SH, Ebrahimi A. Ideological Sublations: Resolution of Dialectic in Population-based Optimization; 2017.

[583] Gandomi AH. Interior search algorithm (ISA): A novel approach for global optimization. ISA Transactions. 2014;53(4); p. 1168–1183.

[584] Hajiaghaei-Keshteli M, Aminnayeri M. Solving the integrated scheduling of production and rail transportation problem by Keshtel algorithm. Applied Soft Computing. 2014;25; p. 184–203.

[585] Jaddi NS, Alvankarian J, Abdullah S. Kidney-inspired algorithm for optimization problems. Communications in Nonlinear Science and Numerical Simulation. 2017;42; p. 358–369.

[586] De Melo VV. Kaizen Programming. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation; 2014. p. 895–902.

[587] Houssein EH, Oliva D, Samee NA, Mahmoud NF, Emam MM. Liver Cancer Algorithm: A novel bio-inspired optimizer. Computers in Biology and Medicine. 2023;165; p. 107389.

[588] Ni L, Ping Y, Yao N, Jiao J, Wang G. Literature Research Optimizer: A New Human-Based Metaheuristic Algorithm for Optimization Problems. Arabian Journal for Science and Engineering. 2024;p. 1–49.

[589] Nishida TY. Membrane Algorithms: Approximate Algorithms for NP-Complete Optimization Problems. In: Applications of Membrane Computing. Springer Berlin Heidelberg; 2006. p. 303–314.

[590] Sadollah A, Bahreininejad A, Eskandar H, Hamdi M. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. Applied Soft Computing. 2013;13(5); p. 2592–2612.

[591] Asil Gharebaghi S, Ardalan Asl M. New meta-heuristic optimization algorithm using neuronal communication. Iran University of Science & Technology. 2017;7(3); p. 413–431.

[592] Khouni SE, Menacer T. Nizar optimization algorithm: A novel metaheuristic algorithm for global optimization and engineering applications. The Journal of Supercomputing. 2024;80(3); p. 3229–3281.

[593] Kaveh A, Akbari H, Hosseini SM. Plasma generation optimization: A new physically-based metaheuristic algorithm for solving constrained optimization problems. Engineering Computations. 2020;38(4); p. 1554–1606.

[594] Chan CY, Xue F, Ip W, Cheung C. A hyper-heuristic inspired by pearl hunting. In: International Conference on Learning and Intelligent Optimization; 2012. p. 349–353.

[595] Savsani P, Savsani V. Passing vehicle search (PVS): A novel metaheuristic algorithm. Applied Mathematical Modelling. 2016;40(5–6); p. 3951–3978.

[596] Jiang Q, Wang L, Hei X, Fei R, Yang D, Zou F, et al. Optimal approximation of stable linear systems with a novel and efficient optimization algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC; 2014. p. 840–844.

[597] Ansótegui C, Pon J, Sellmann M, Tierney K. Reactive Dialectic Search Portfolios for MaxSAT. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence; 2017. p. 765—-772.

[598] Dorigo M, Stützle T. Ant Colony Optimization. MIT Press; 2004.

[599] Sörensen K. Metaheuristics - the metaphor exposed. International Transactions in Operational Research. 2015;22; p. 3–18.

[600] García-Martínez C, Gutiérrez PD, Molina D, Lozano M, Herrera F. Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness. Soft Computing. 2017;21(19); p. 5573–5583.

[601] Liao T, Molina D, Stützle T. Performance evaluation of automatically tuned continuous optimizers on different benchmark sets. Applied Soft Computing. 2015;27; p. 490–503.

[602] Bosman PAN, Gallagher M. The importance of implementation details and parameter settings in black-box optimization: a case study on Gaussian estimation-of-distribution algorithms and circles-in-a-square packing problems. Soft Computing. 2018;22(4); p. 1209–1223.

[603] Biedrzycki R. On Equivalence of Algorithm's Implementations: The CMA-ES Algorithm and Its Five Implementations. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion; 2019. p. 247–248.

[604] Liefooghe A, Jourdan L, Talbi EG. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research. 2011;209(2); p. 104–112.

[605] Durillo JJ, Nebro AJ. jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software. 2011;42(10); p. 760–771.

[606] Vrbančič G, Brezočnik L, Mlakar U, Fister D, Fister Jr I. NiaPy: Python microframework for building nature-inspired algorithms. Journal of Open Source Software. 2018;3(23); p. 613.

[607] Benítez-Hidalgo A, Nebro AJ, García-Nieto J, Oregi I, Ser JD. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. Swarm and Evolutionary Computation. 2019;51; p. 100598.

[608] Tian Y, Cheng R, Zhang X, Jin Y. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. IEEE Computational Intelligence Magazine. 2017;12(4); p. 73–87.

[609] Valadi J, Siarry P. Applications of metaheuristics in process engineering. vol. 31. Springer; 2014.

[610] Gandomi AH, Yang XS, Talatahari S, Alavi AH. Metaheuristic Applications in Structures and Infrastructures. New York: Springer; 2013.

[611] Griffis SE, Bell JE, Closs DJ. Metaheuristics in logistics and supply chain management. Journal of Business Logistics. 2012;33(2); p. 90–106.

[612] Starodubcev NO, Nikitin NO, Kalyuzhnaya AV. Surrogate-Assisted Evolutionary Generative Design Of Breakwaters Using Deep Convolutional Networks. In: 2022 IEEE Congress on Evolutionary Computation (CEC); 2022. p. 1–8.

[613] Lohn JD, Hornby GS, Linden DS. An evolved antenna for deployment on nasa's space technology 5 mission. Genetic Programming Theory and Practice II. 2005;p. 301–315.

[614] Schmidt M, Lipson H. Symbolic regression of implicit equations. In: Genetic programming theory and practice VII. Springer; 2009. p. 73–85.

[615] Li N, Ma L, Xing T, Yu G, Wang C, Wen Y, et al. Automatic design of machine learning via evolutionary computation: A survey. Applied Soft Computing. 2023;143; p. 110412.

[616] Song H, Triguero I, Özcan E. A review on the self and dual interactions between machine learning and optimisation. Progress in Artif Intell. 2019;8(2); p. 143–165.

[617] Real E, Liang C, So D, Le Q. AutoML-Zero: Evolving Machine Learning Algorithms From Scratch. In: Proceedings of the 37th International Conference on Machine Learning. vol. 119; 2020. p. 8007–8019.

[618] Liu Y, Sun Y, Xue B, Zhang M, Yen GG, Tan KC. A Survey on Evolutionary Neural Architecture Search. IEEE Transactions on Neural Networks and Learning Systems. 2023;34(2); p. 550–570.

[619] Dufourq E, Bassett BA. EDEN: Evolutionary deep networks for efficient machine learning. In: Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech); 2017. p. 110–115.

[620] Assunçao F, Lourenço N, Machado P, Ribeiro B. DENSER: deep evolutionary network structured representation. Genetic Programming and Evolvable Machines. 2019;20; p. 5–35.

[621] Wang R, Lehman J, Clune J, Stanley KO. Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions; 2019.

[622] Akiba T, et al.. Evolutionary Optimization of Model Merging Recipes; 2024.

[623] He X, Zhao K, Chu X. AutoML: A survey of the state-of-the-art. Knowledge-Based Systems. 2021;212; p. 106622.

[624] Ma YJ, Liang W, Wang G, Huang DA, Bastani O, Jayaraman D, et al.. Eureka: Human-Level Reward Design via Coding Large Language Models; 2023.

[625] Zhao H, Ning X, Liu X, Wang C, Liu J. What makes evolutionary multi-task optimization better: A comprehensive survey. Appl Soft Comput. 2023 Sep;145; p. 110545.

[626] López-ibánez M, Branke J, Paquete L. Reproducibility in Evolutionary Computation. ACM Transactions on Evolutionary Learning and Optimization. 2021;1(4).

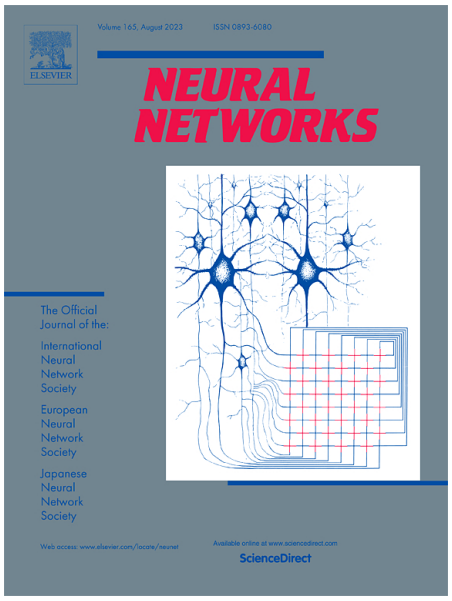
[627] Wu X, hao Wu S, Wu J, Feng L, Tan KC. Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap; 2024.

[628] Triguero I, Molina D, Poyatos J, Del Ser J, Herrera F. General Purpose Artificial Intelligence Systems (GPAIS): Properties, definition, taxonomy, societal implications and responsible governance. Information Fusion. 2024;103; p. 102135.

[629] Guo Q, et al. Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers. In: The Twelfth International Conference on Learning Representations; 2024. .

[630] Chen S, Chen S, Hou W, Ding W, You X. EGANS: Evolutionary Generative Adversarial Network Search for Zero-Shot Learning. IEEE Transactions on Evolutionary Computation. 2023;p. In Press.

2 EvoPruneDeepTL: An evolutionary pruning model for transfer learning based deep neural networks



- Journal: Neural Networks
- JCR Impact Factor: 7.8
- Rank: 28/145
- Quartile: Q1
- Category: Computer Science, Artificial Intelligence
- Status: Published

Ref.: Poyatos, J., Molina, D., Martinez, A. D., Del Ser, J., & Herrera, F. (2023). EvoPruneDeepTL: An evolutionary pruning model for transfer learning based deep neural networks. Neural Networks, 158, 59-82. DOI: https://doi.org/10.1016/j.neunet.2022.10.011

EvoPruneDeepTL: An Evolutionary Pruning Model for Transfer Learning based Deep Neural Networks

Javier Poyatos[a], Daniel Molina[a,*], Aritz D. Martinez[b], Javier Del Ser[b,c], Francisco Herrera[a,d]

[a]Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, Granada, 18071, Spain
[b]TECNALIA, Basque Research & Technology Alliance (BRTA), Derio, 48160, Spain
[c]University of the Basque Country (UPV/EHU), Bilbao, 48013, Spain
[d]Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 21589, Saudi Arabia

Abstract

In recent years, Deep Learning models have shown a great performance in complex optimization problems. They generally require large training datasets, which is a limitation in most practical cases. Transfer learning allows importing the first layers of a pre-trained architecture and connecting them to fully-connected layers to adapt them to a new problem. Consequently, the configuration of the these layers becomes crucial for the performance of the model. Unfortunately, the optimization of these models is usually a computationally demanding task. One strategy to optimize Deep Learning models is the pruning scheme. Pruning methods are focused on reducing the complexity of the network, assuming an expected performance penalty of the model once pruned. However, the pruning could potentially be used to improve the performance, using an optimization algorithm to identify and eventually remove unnecessary connections among neurons. This work proposes EvoPruneDeepTL, an evolutionary pruning model for Transfer Learning based Deep Neural Networks which replaces the last fully-connected layers with sparse layers optimized by a genetic algorithm. Depending on its solution encoding strategy, our proposed model can either perform optimized pruning or feature selection over the densely connected part of the neural network. We carry out different experiments with several datasets to assess the benefits of our proposal. Results show the contribution of EvoPruneDeepTL and feature selection to the overall computational efficiency of the network as a result of the optimization process. In particular, the accuracy is improved, reducing at the same time the number of active neurons in the final layers.

Keywords- Deep Learning, Evolutionary Algorithms, Pruning, Feature Selection, Transfer Learning

*Corresponding author
Email addresses: jpoyatosamador@ugr.es (Javier Poyatos), dmolina@decsai.ugr.es (Daniel Molina), aritz.martinez@tecnalia.com (Aritz D. Martinez), javier.delser@tecnalia.com (Javier Del Ser), herrera@decsai.ugr.es (Francisco Herrera)

1. Introduction

Deep Learning (DL) (Goodfellow et al., 2016) is one of the most attractive research areas in machine learning in recent times, due to the great results offered by such models in a plethora of applications. DL architectures are successfully used in many problems, like audio classification (Lee et al., 2009, December), audio recognition (Noda et al., 2015), object detection (Zhou et al., 2017, May), image classification for medical analysis (Muhammad et al., 2021) or vehicular perception (Muhammad et al., 2020).

Convolutional Neural Networks (CNNs) (Lecun et al., 1998) constitute the state-of-the art in image classification (Sultana et al., 2018, November). CNNs include two parts, the first part is actually a feature extractor based on convolution and pooling operations. The second part usually contains one or more fully connected layers. In these fully-connected layers, the neuron of each layer is connected to all the neurons of the previous layer, which generates a large number of weights to be trained. The design of an appropriate network for each problem is a requirement in order to obtain a good performance. The training process of a DL architecture is frequently time-consuming. Complexity reduction maintaining the performance is an important challenge in DL, currently attracting significant attention in the community. Transfer Learning (TL) (Weiss et al., 2016) is usually considered the alternative. It is very common to use a DL model with fixed and pre-trained weights in the convolutional layers with a dataset (like ImageNet (Krizhevsky et al., 2012, December)) and then add and train several layers, named fully-connected layers, to adapt the network to a different classification problem (Shin et al., 2016; Khan et al., 2019; [dataset] Gómez-Ríos et al., 2019).

The architecture of fully-connected layers used for the problem is a critical decision, and its design is still an open issue in terms of the number of layers and neurons per layer (Liu et al., 2017). There are general guidelines based on the experience working with these layers, rather than rules to follow for the configuration of them. Therefore, any kind of optimization in them could provide a benefit in terms of model complexity and performance. The pruning approaches follow the key idea of reducing the complexity of the model, which creates new networks with less computational cost for training. This idea is followed in (Frankle & Carbin, 2019, May), which also shows that, in the end, the accuracy can also improve as a result of pruning.

Pruning is interpreted as removing unnecessary connections from the model, but learning which connections are the fittest to improve the performance of the model is the key point. In fact, the selection of the best *features* for the problem is known as Feature Selection (FS) (Iguyon & Elisseeff, 2003). In our case, TL allows the extraction of the features of the input data of the DL model. These features are the input of the fully-connected layers that will be trained and, as a result of that, will largely affect the performance of the network. Nonetheless, in many cases, the problem that is formulated to learn these features is usually different, sometimes more complex, than the one at hand and, therefore, not all the learned patterns would be required. For that reason, FS gives rise to an interesting option to select and retain the subset of all features that lead to an improved performance of the model (Yildirim et al., 2018).

In pruning scenarios, the main aim of most of the traditional pruning techniques mainly aim at reducing the number of trainable parameters of the network, at the cost of a lower performance. They seek to control the performance degradation resulting from the process, but it is not their priority. Furthermore, they locally optimize parts of the network rather than searching for globally optimal pruning policies, yielding usually suboptimal pruned subnetworks with a lower performance. Another disadvantage of these pruning proposals is the fact that, as the pruning affect all layers, the complete network must be trained again, hence obtaining no advantages from the TL process. It could be useful to have a pruning technique that prioritizes results over complexity reduction, targeting a global performance improvement of the network while reducing its complexity.

Transforming the fully-connected layers into a sparse representation, in which each connection could be active or inactive, could be used to prune neural networks. Following this approach, both pruning and FS can be seen as optimization problems, in which the target is to obtain the active set of connections that produce the best performance. This optimization problem can be globally tackled by optimization algorithms like Evolutionary Algorithms (Back et al., 1997) (EAs). They have been successfully applied to many complex optimization problems. Even though they cannot guarantee the achievement of the optimum for the problem at hand, they obtain good results with limited resources and reasonable processing time. Another advantage is their versatility: several of them, like genetic algorithms (Goldberg, 1989) (GAs) allow

optimizing solutions with different representations (Chambers, 2000). The spectrum of problems in which EAs can be used is very wide. EAs have been traditionally applied to optimize neural networks (Iba, 2018), but their usage in DL networks to improve DL networks (Martinez et al., 2021), to train them (Mohapatra et al., 2022), and to create new DL networks from scratch (Elsken et al., 2019) is more recent. The use of EA's is mainly oriented towards optimizing a complete network. However, in this paper, our aim is to adapt the fully-connected layers (the only trained for the problem to solve using TL) to improve the accuracy in the predictions, together with the complexity reduction. Our main hypothesis is the convenience of use of EAs to prune the fully-connected layers via a sparse representation.

We propose an evolutionary pruning model based on TL for deep neural networks, Evolutionary Pruning for Deep Transfer Learning (EvoPruneDeepTL). EvoPruneDeepTL can be applied to a DL model that resorts to TL to tackle a new task. EvoPruneDeepTL combines sparse layers and EA, consequently, neurons in such layers are pruned to adapt their sparsity pattern to the addressed problem. EvoPruneDeepTL is able to efficiently explore the neuron search space (to discover coarsely grained solutions) or, alternatively, in the connection search domain (fine-grained solutions).

An important aspect to analyze in EvoPruneDeepTL is that one of its solution encoding schemes effectively leads to a feature selection mechanism, in which we deactivate the extracted features and the EA evolves these features to learn which ones fit best as predictors for the given problem.

EvoPruneDeepTL' goals include flexibility and adaptability. EvoPruneDeepTL has been designed to be flexible, and the automatic configuration of the network can be applied to different pre-trained networks, used as feature extractors, and different fully-connected layers. This make our proposal capable of tackling different problems. The removal of connections that EvoPruneDeepTL performs allows the model to be adaptable to the specific dataset to be modeled. Thus, when the dataset suffers a change, the resultant configuration will also be adapted to the new circumstances.

To assess the performance of EvoPruneDeepTL, we have conducted an extensive experimentation that leads to several valuable insights. To begin with, experimental results showcase the behavior and effectiveness of EvoPruneDeepTL in terms of precision and in terms of reduction of the complexity of the network. Thanks to the flexibility of EvoPruneDeepTL, it is applied to perform either pruning or FS. Both cases improve the accuracy of the network when the comparison is made against reference models and CNN pruning methods from the literature. Moreover, in most cases, the FS scheme achieves a better performance than the pruning scheme in terms of the accuracy of the network. Furthermore, the network pruned by the FS scheme also achieves a significantly reduced number of connections in its fully connected part, contributing to the computational efficiency of the network. We have also included several experiments showing the flexibility of the model, both changing the feature extractor and showing how changes in the dataset implies a modification in the final configuration obtained by EvoPruneDeepTL. In short, this extensive experimentation is used to provide answer to the following six questions as the thread running through this experimental study:

(**RQ1**) Which is the performance of EvoPruneDeepTL against fully-connected models?

(**RQ2**) Which would be better, to remove neurons or connections?

(**RQ3**) Which is the performance of EvoPruneDeepTL when compared to other efficient pruning methods?

(**RQ4**) Which would be better, the use of pruning of fully-connected layers or Feature Selection?

(**RQ5**) How does EvoPruneDeepTL perform when applied to different pre-trained networks?

(**RQ6**) Can EvoPruneDeepTL adapt efficiently their pruned knowledge to changes in the modeling task, showing robustness?

The rest of the article is structured as follows: Section 2 exposes related work to our proposal present in the literature. Section 3 shows the details of the proposed EvoPruneDeepTL model. Section 4 presents our experimental framework. In Section 5, we show and discuss the EvoPruneDeepTL's results of the experiments of pruning, feature selection and against efficient CNN pruning methods of the literature. Moreover, EvoPruneDeepTL is tested with different extractor features and with different variations of datasets in this

section. Section 6 follows by summarizing the advantages and drawbacks of our proposal when compared to other pruning approaches. Finally, Section 7 draws the main conclusions stemming from our work, and outlines future work departing from our findings.

2. Related work

The purpose of this section is to make a brief review of contributions to the literature that link to the key elements of our study: Transfer Learning (Subsection 2.1), Neural Architecture Search (Subsection 2.2), CNN pruning (Subsection 2.3), Evolutionary Algorithms (Subsection 2.4) and Feature Selection with Deep Learning (Subsection 2.5).

2.1. Transfer Learning

TL (Pan & Yang, 2010) is a DL mechanism encompassing a broad family of techniques (Tan et al., 2018, October). Arguably, the most straightforward method when dealing with neural networks is *Network-based deep transfer learning*, in which a previous network structure with pre-trained parameters in a similar problem is used. It offers good results by the behavior of DL models, in which first layers detect useful features on the images, and later layers strongly depend on the chosen dataset and task. As finding these standard features on the first layers seems very common regardless of the natural image datasets, its trained values can be used for different problems (Yosinski et al., 2014, December). Training DL models from scratch is usually time-consuming due to the great amount of data in most cases. TL gives some benefits which make it a good option for DL: reduction of time needed for training (Sa et al., 2016), better performance of the model and less need of data.

TL has been applied to several real-world applications, such as sound detection (Jung et al., 2019, May) or coral reef classification (Gómez-Ríos et al., 2019). Moreover, in (Tajbakhsh et al., 2016) two different approaches for TL are discussed: fine-tuning or full training. They demonstrated that, for medical reasons, a pre-trained CNN with adequate fine-tuning performed better in terms of accuracy than a CNN trained from scratch. Another approach of TL is presented in (Mehdipour Ghazi et al., 2017), in which an optimization of TL parameters for plant identification is proposed.

There are different deep neural networks proposed in the literature. One of the most popular is ResNet, which uses residual learning to improve the training process, obtaining better performance than other models (He et al., 2016, June). ResNet models are characterized by the use of deeper neural networks without loss of information due to their architecture. Different ResNet models with TL have been used in several applications (Scott et al., 2017), such as medical classification like pulmonary nodule (Nibali et al., 2017) and diabetic retinopathy classification(Wan et al., 2018). Moreover, other networks have shown great performance when used with TL, such as DenseNet (Huang et al., 2017, July) and VGG (Simonyan & Zisserman, 2015, May). An example of DenseNet with TL is presented in (Aneja & Aneja, 2019, July), which shows that this network architecture is able to achieve a great result for the task at hand when combined with TL. Lastly, VGG has also shown an outstanding performance when it is used in combination with TL. An example is presented in (Wen et al., 2019, May) in which a pre-trained VGG-19 network is used to solve a fault diagnosis problem.

2.2. Neural Architecture Search

The appropriate design of a neural network is a key point to solve DL problems. Nevertheless, finding the best architecture that optimally fits the data and, as a result of that, gives the best outcome for the problem is extremely difficult. Recently, the term Neural Architecture Search, NAS, has obtained a great importance in this field. The objective of NAS is the automatic search for the best design of a NN to solve the problem at hand.

The first work in this field is presented in (Stanley & Miikkulainen, 2002) in the beginning of this century, in which they demonstrate the effectiveness of a GA to evolve topologies of NN.

In (Zoph et al., 2018, June), the authors design the NASNet architecture, a new search space to look for the best topology for the tackled problem. Moreover, in (Liu et al., 2018, September) the authors propose

to search for structures in increasing order of their level of complexity, while learning a surrogate model to guide the search through structure space.

NAS methods usually rely on Reinforcement Learning, RL, and EA, like (Zoph et al., 2018, June) or (Liu et al., 2018, April), in which the authors explore the search space using a hierarchical genetic representation. Another example of RL for NAS is shown in (Kokiopoulou et al., 2020, August). The authors propose a novel method that, by sharing information on multiple tasks, is able to efficiently search for architectures.

NAS can also be viewed as a multi-objective problem. Among these methods, one of them is presented in (Elsken et al., 2019, May), in which the authors propose a multi-objective for NAS that allows approximating the entire Pareto-front of architectures. Another example is Neural Architecture Transfer (Lu et al., 2021), that allows to overcome a common limitation of NAS, that is requiring one complete search for each deployment specification of hardware or objective. They use an integrated online transfer learning and a many-objective evolutionary search procedure.

Recently, one of the most well-known multi-objective EA, NSGA-II, has been used for NAS (Lu et al., 2019, July), called NSGA-Net. This novel proposal looks for the best architecture through a three-step search: an initialization step from hand-crafted architectures, an exploration step that performs the EA operators to create new architectures, and finally an exploitation step that utilizes the knowledge stored in the history of all the evaluated architectures in the form of a Bayesian Network.

Lastly, there are more advanced techniques of NAS and EA given by (Real et al., 2019, January), in which a new model for evolving a classifier is presented, and by (Real et al., 2020, July), in which the authors propose AutoML-Zero, an evolutionary search to build a model from scratch (with low-level primitives for feature combination and neuron training) which is able to get a great performance over the addressed problem.

2.3. CNN Pruning

The main reason to optimize the architecture of a deep neural network is to reduce its complexity. That reduction can be done in different ways (Long et al., 2019a). One of them is by designing compact models from scratch instead of resorting to architectures comprising multiple layers. Another strategy is via weights-sharing (Ullrich et al., 2017, April). An alternative method to reduce the complexity of DL models is low-rank factorization (Long et al., 2019b), based on a matrix decomposition to convolutional layers to estimate parameters. However, one of the most popular is Network Pruning. The objective of pruning is to remove unnecessary parameters from a neural network, so that they do not participate during training and/or inference. It can be done in the convolutional phase on the channels, kernels and weights or even in the fully connected phase on the neurons. In (Masson et al., 2021) they show a classification of pruning methods for channels. They categorize the pruning methods for channel reduction, and they also specify the criteria used to select these channels: based on weights or based on feature maps.

We have seen that network pruning has achieved a great importance in the literature as many researchers have applied different techniques to simplify a CNN using a pruning scheme. In (Liu et al., 2019, May) they classify the pruning methods in unstructured and structured pruning, and make a review of all the state-of-art structured pruning methods. Unstructured pruning methods remove weights without following any order. For the structured methods, there are some rules or even constraints which define how the pruning is done (Anwar et al., 2017). Typically, the pruned layers appertain to the convolutional phase (Luo et al., 2017, October). In our proposal, we instead apply a structured pruning scheme to the fully-connected layers.

Among pruning methods, the value-based *weight pruning* (Han et al., 2015, Dececember) and *neuron pruning* (Srinivas & Babu, 2015, September) have arisen as the most used, particularly due to their simplicity. The logic behind this pruning methods is straightforward: a certain amount (%) of the weights or neurons that contribute less to the final trained model are removed from the architecture. This makes the network quicker to perform inference and endows it with better generalization capabilities. However, multiple pruning and retraining steps demonstrated that it is possible to recover fully or partially the knowledge lost in the pruning phase. Further along the series of pruning approaches published to date, *Polynomial Decay* (Zhu & Gupta, 2018, April) is a scheduled pruning method that considers that a higher amount of weights can be pruned in early stages of pruning, while systematically less amount of weights should be pruned in late

stages. Between pruning steps, the network is retrained for some epochs. An implementation of the discussed methods can be found for Tensorflow. [1]

Pruning a CNN model reduces its complexity, but sometimes leads to a decrease of the performance of the model, although there are some proposals that reduce the complexity of the model with no loss of accuracy (Han et al., 2016, May).

Pruning a neural network can be conceived as an optimization method in which we start from the original vector, and connections/neurons are decision variables whose value is evolved towards optimizing a given objective. In this context of evolution of neural networks, evolutionary algorithms for evolving DL architectures have been applied (Iba, 2018). While this combination of EA and DL models seemed to be a great scheme for the optimization of DL models, especially for CNN network, the optimization of DL models is still an open problem (Liu et al., 2017). Many proposals have been published about this problem like in (Martinez et al., 2021), where they make a review of proposals using EAs for optimizing DL models, prescribing challenges and future trends to effectively leverage the synergy between these two areas.

Researchers have presented a great variety of proposals about the optimization of DL models using EA, most commonly for CNN. In (Martín et al., 2018), the authors developed EvoDeep, an EA with specific mutation and crossover operators to automatically create DL models from scratch. Moreover, in (Real et al., 2017, August) a novel evolution approach to evolve CNN models using a GA was proposed. Another example of the optimization of CNN was developed in (Assunção et al., 2019), in which a GA was presented for the optimization of the topology and parameters of the CNN.

In our proposal, we improve the performance of the models using a TL approach to extract the features of the images and apply a reduction of the fully-connected layer using a GA to optimize a sparse layer.

2.4. Evolutionary Algorithms for CNN Pruning

In the previous section, several works of CNN pruning have been presented, but none of them use an EA to prune. In this section, we mention some studies present in the literature which have used an EA in order to prune a CNN model. To begin with, in (Liu et al., 2017, February), the authors propose a sparse approach to reduce CNN complexity. EAs are also a good way to prune CNN. In (Mantzaris et al., 2011), a first attempt of pruning and GA is proposed for a medical application. They use a GA to search for redundancy factors in a neural network. Moreover, in (Samala et al., 2018) another EA is presented to prune deep CNN for breast cancer diagnosis in digital breast tomosynthesis. A combined approach of EA and sparse is proposed by Wang et al. (Wang et al., 2020), in which a GA and sparse learning are applied to a scheme of network channel pruning in the convolutional scheme of the CNN. For pruning CNN, not only GAs but also other algorithms are used, like Differential Evolution (DE). In (Salehinejad & Valaee, 2021) the authors propose to use a Differential Evolution algorithm to prune the convolutional phase and the fully-connected phase of some Deep CNN, obtaining a reduction of the model but a small decrease of its performance.

However, all previous works are focused on reducing the complexity, using the EA to reduce the accuracy loss of the pruned network. Also, many of them try to reduce the whole model, changing the complete architecture and making the pre-trained values unusable. The re-training of the network may be a time-consuming task, so we assume that TL is useful in this context. We therefore maintain the original architecture with pre-trained values. Our model focuses on improving the performance of the model by pruning connections of the fully-connected layers using a GA to evolve the connections. In this environment, the search space of the GA is narrower and a faster convergence of the algorithm may be reached.

In addition to that, in the field of neural architecture search, more advanced techniques have been developed. Among them, in Section 2.2, either (Real et al., 2019, January) and (Real et al., 2020, July) have been commented. Nonetheless, they also have a great relevance in this section. The first one evolves a classifier, whereas in the second one, the authors develop an evolutionary search to build a model from scratch.

[1] https://www.tensorflow.org/model_optimization/guide/pruning, Tensorflow Pruning. Last access: 28/01/2022

2.5. Feature Selection and Deep Learning

One of the advantages of using TL is reducing the required time to train a DL architecture. Nonetheless, the result of this process may lead to recognize patterns that are not useful to address the problem at hand. For that reason, once TL is applied, a FS process to obtain the best features might lead to a better performance of the neural network (Roy et al., 2015, July).

An example of this process is presented in an arrhythmia detection task addressed in (Yildirim et al., 2018), in which the authors propose a mechanism based on feature extraction and selection to improve and ultimately obtain one of the best results for this problem. In relation to medical problems, the combination of FS and DL is also used in cancer diagnosis and digital breast tomosynthesis. In (Samala et al., 2018) they use a TL approach and then a FS process followed by an evolution through a GA that leads to a reduced network with the same performance. Another example is described for remote sensing scene classification, in which the FS makes an impact to improve the performance of the neural network models (Zou et al., 2015), as the authors formulate the FS problem as a feature reconstruction problem. Their iterative method selects the best features to solve this problem as the discriminative features.

In our proposal, if we assume that TL is applied and we only have one fully-connected layer, then the pruning is made in relation to the extracted features of the network and, therefore, we are making a selection of the features that adjust at best to the tackled problem.

3. Evolutionary Pruning for Deep Transfer Learning

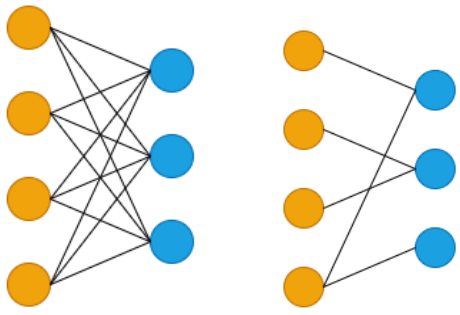
This section describes EvoPruneDeepTL, which is a model that replaces fully-connected layers with sparse layers which are being evolved using a genetic algorithm in a TL approach. Subsection 3.1 gives a notion of the concept of sparse layer and the description of EvoPruneDeepTL. In Subsection 3.2, we define the evolutionary components of EvoPruneDeepTL. The description of the process of creating the network and how the pruning is made is shown in Subsection 3.3.

3.1. Global scheme of Evolutionary Pruning for Deep Transfer Learning

In a fully-connected layer, all neurons of each side are connected. Sometimes, all these connections may not be necessary, and the learning process can be reduced. For that reason, the fully-connected layer can be replaced by a sparse layer, in which some connections are eliminated.

In this work, our goal is to improve the performance of the neural network and, at the same time, to decrease the maximum number of connections or neurons. To this end, we use a sparse layer, which is composed of a subset of all connections of a fully-connected layer.

Fig. 1a shows the fully-connected network architecture, while Fig. 1b represents the sparsely connected architecture with a connection matrix of 4×3 because we have 3 classes (blue circles) and 4 neurons of the previous layer.



(a) Fully-Connected Layer (b) Sparse Layer

Figure 1: Representation of both architectures

In this section, we discuss the basic notions of pruning and sparse layers. Moreover, the encoding strategies of EvoPruneDeepTL are described, together with the decoding process of the chromosome encoding the pruning pattern (genotype) that yields the pruned sparse layer(s) (phenotype). Nonetheless, for the sake of a clear vision of EvoPruneDeepTL, Fig. 2 shows a diagram that exposes its general components. We next complement the detailed description provided in the following subsections with a short, albeit illustrative introduction to the key parts and overall workflow of EvoPruneDeepTL:

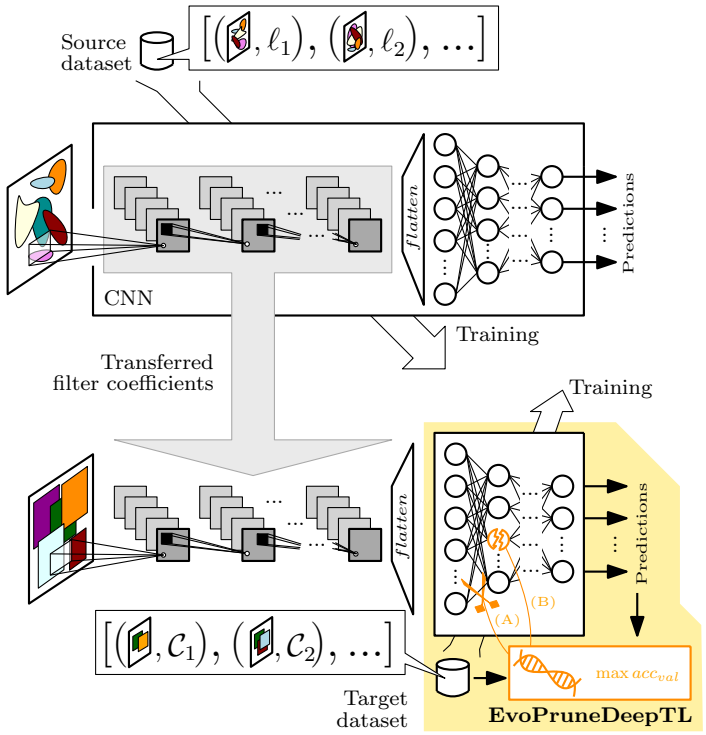


Figure 2: Diagram of EvoPruneDeepTL.

First, the TL process departs from a source dataset modeled by a CNN model, which adjusts the value of its coefficients by means of its learning algorithm. Then, those parameters are transferred to another network aimed to model a target task. This implicitly assumes that both tasks are correlated with each other, such that the knowledge delivered from the source to the target task via the transferred network weights can positively contribute to the learning process of the target task. These weights are kept fixed, *frozen*, in this study. Then, EvoPruneDeepTL specializes the fully-connected part of the neural network of the target task by resorting to a GA. This metaheuristic wrapper prunes unnecessary neurons of these layers driven by the improvement of a performance measure (e.g. accuracy). The outcome of the process is a pruned network with a potentially improved accuracy by virtue of an evolved pruning mask.

In this study, we propose a novel method to prune the neurons, that considers the removal of both single connections and groups of connections of the input connections of a specific neuron, as can be observed in Fig. 3. Fig 3b shows a sparse layer that leads to the encoding strategy used in this work. This encoding, which is represented by the chromosome of the GA, is required to know exactly which connections are removed.

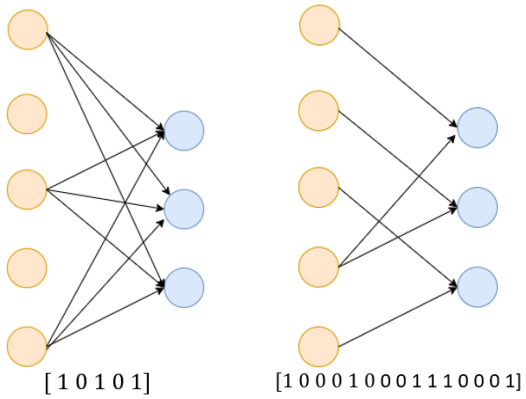
EvoPruneDeepTL model utilizes a GA designed to evolve the connections of a sparse layer. The GA takes each individual as a mask for the neural network and creates a sparse layer activating from the mask. This evolved mask gives rise to a pruned neural network suitable for the problem under consideration.

The evolution of the connections is performed using both methods, either by groups of connections or by single connections. The genome representation of each chromosome of the GA is binary-coded and represents

the active *neurons* or the active *connections*. The GA evolves the configuration of the network towards its best pruned variant in terms of accuracy. Next, we describe both encoding strategies:

- **Neurons**: each gene of the chromosome represents the number of active neurons. A value 1 in position i means that the neuron i is active, and a 0 that is inactive. A non-active neuron implies that all the input connections are removed both in training and inference times. The length of the chromosome in this case is the number of neurons of the sparse model.
- **Connections**: each gene represents the connection between the layers. The interpretation of the binary values is as follows: if a gene is 1, the connection between the corresponding layers exists, otherwise, that connection does not exist. Therefore, the length of the chromosome is the maximum number of connections, noted as $D = D_1 \times D_2$, where D_1 is the number of neurons in the previous layer, and D_2 is the number of neurons in that layer.

An example of both encoding strategies is shown in Fig. 3. In both cases, the pruned connections are from the input on the layer, i.e. the right layer. The left image shows a representation of neuron-wise encoding, in which a group of neurons is selected to be active and the rest are pruned. The right image depicts how single connections are pruned.



(a) Neuron Encoding (b) Single Connection Encoding

Figure 3: Representation of encoding strategies

3.2. Evolutionary components of EvoPruneDeepTL

In this subsection, we introduce the evolutionary components of EvoPruneDeepTL. It is a steady-state genetic algorithm, which means that two new individuals, called offsprings, are created in each generation, for the previous mentioned encoding strategies (neuron encoding vs single connection encoding, Fig 3): in each iteration two individuals are selected and crossed, producing two offsprings that could also be mutated. The offspring candidates are introduced in the population only if they improve the worst candidates in the population, replacing them.

As previously mentioned, in EvoPruneDeepTL each chromosome is a binary array and each gene represents a connection between two layers. Each generation follows the classical scheme of selection, crossover, mutation and replacement. The best solutions found during the evolutionary search are kept in a population of individuals. Next, we describe the different components:

Selection: the implemented selection operator is Negative Assorting Mating (NAM) (Fernandes & Rosa, 2001, May). The first parent is picked uniformly at random, while the second parent is selected between three possible candidates. These candidates are also picked uniformly at random from the population. The candidate with higher Hamming distance from the first parent is chosen as the second parent, thereby ensuring that the recombined parents are diverse. This selection method allows for a higher degree of exploration of the search space.

Crossover: EvoPruneDeepTL uses the uniform crossover operator shown in Expression 1. Given two parents **P** and **Q**, where $\mathbf{P} = \{p_i\}_{i=1}^{D}$ and $\mathbf{Q} = \{q_i\}_{i=1}^{D}$. Then two offsprings **P'** $= \{p'_i\}_{i=1}^{D}$ and **Q'** $= \{q'_i\}_{i=1}^{D}$ are created following the equations:

$$
\begin{aligned}
pi' &= \begin{cases} p_i & \text{if } r \leq 0.5 \\ q_i & \text{otherwise} \end{cases} \\
q'_i &= \begin{cases} q_i & \text{if } r \leq 0.5 \\ p_i & \text{otherwise} \end{cases}
\end{aligned}
\tag{1}
$$

where r is the realization of a continuous random variable with support over the range $[0.0, 1.0]$. This operator takes two parents **P** and **Q** of length D and creates two new offspring **P'** and **Q'** of the same size. Each new offspring is composed of the parents genes. Each gene (position of the new array) is set equal to the gene of the first or second parent. This process is repeated until the whole offspring is composed.

Mutation: EvoPruneDeepTL adopts the so-called single point mutation. A mutation probability for each individual is defined by p_{mut}. Then, a gene of that individual is uniformly randomly selected and its bit is flipped, i.e., if the mutation is performed, then that neuron or connection changes its value, which implies that the connection or the neurons is activated or deactivated. In this operator, p_{mut} is the value that establishes the probability that a mutation is performed.

Replacement Strategy: at the end of every generation, the two offsprings resulting from the crossover and mutation operators compete against the worst two elements. As a result, the population is updated with the best two individuals among them, i.e. those whose fitness value is better. EvoPruneDeepTL maintains a pool of four individuals: two offsprings and the two worst individuals selected from the population. Then, the best two of them are in the new population. The criterion to select the best two is based on the fitness as the best of them are selected. In case of same values, the individuals with fewer active neurons/connections are those selected to be retained in the new population.

Initialization: the genes composing the individuals are initialized to 0 or 1 as per the following probabilistic condition with a p_{one} probability:

$$
I_i = \begin{cases} 1 & \text{if } r \leq p_{one} \\ 0 & \text{otherwise} \end{cases}
\tag{2}
$$

where r is the realization of a uniform continuous random variable with support $[0.0, 1.0]$.

Evaluation of individuals: the fitness value of every individual is given by the accuracy over a test dataset of the neural network pruned as per the decoded individual, and trained over the training dataset of the task at hand. Each individual, named p, is decoded to yield a sparse neural network, which we hereafter refer to $SparseNet_p$. Then this network is trained as previously commented over the train dataset, giving the $TrainedSparseNet_p$ network. Lastly, the test dataset is evaluated in this network, producing the fitness of the individual, which we call $ChildFitness_p$.

Algorithm 1 shows the pseudocode of EvoPruneDeepTL. First, we need to understand what EvoPruneDeepTL requires to start its evolution process, and what results from this process. The input of EvoPruneDeepTL is determined by:

- Dataset and task to be modeled.
- Configuration of the GA: parameters needed for the algorithm.
- Configuration of the network: parameters needed for the network.
- Feature extractor: a pre-trained neural network used for feature extraction and TL, e.g., ResNet-50 trained over Imagenet or any other available architecture alike.

The algorithm starts by initializing the individuals of the population (line 1) using the previous operator and then evaluating them (line 2). The evolutionary process is performed in lines 3-15. Two parents are selected using the NAM operator (line 4) and then the two offsprings are generated using the crossover

operator (line 5). If the mutation condition is met, then mutation is performed (lines 6-8). The child population is now evaluated (lines 9-14). The evaluation is held over three steps, in which each individual is decoded (variable SparseNet$_p$ in line 10), and then the network created with its configuration is trained (named TrainedSparseNet$_p$ in line 11) using the train dataset and then evaluated (called ChildFitness$_p$ in line 12) over the test dataset. Lastly, the replacement strategy is triggered (line 15). The stopping criterion is the evaluation of a maximum number of networks.

Algorithm 1: EvoPruneDeepTL

Input : Dataset, configuration of the GA, configuration of the network and feature extractor
Output: Evolved pruned network

```
1  Initialization of individuals of the population using the initialization operator;
2  Evaluation of the initial population (see lines 9-14);
3  while evaluations < max_evals do
4  |  Parent selection using NAM operator;
5  |  Generate offsprings using crossover operator;
6  |  if rand() < p_mut then
7  |  |  Perform mutation using mutation operator;
8  |  end
9  |  for each child p in children population do
10 |  |  SparseNet_p ← Create sparse network using the decoded individual of the population;
11 |  |  TrainedSparseNet_p ← Train SparseNet_p using train dataset;
12 |  |  ChildFitness_p ← Accuracy of TrainedSparseNet_p evaluated in test dataset;
13 |  |  evaluations+=1;
14 |  end
15 |  Replacement Strategy: child population vs worst individuals of population;
16 end
```

3.3. EvoPruneDeepTL Network

This subsection is devised to fully understand the components associated with the networks that involve EvoPruneDeepTL. EvoPruneDeepTL stands for the usage of transfer learning, which means that the convolutional phase before the fully-connected layers is imported from other pre-trained model. Thus, the chosen CNN works as a feature extractor, i.e, obtains the main feature or characteristics for the task at hand. In our study, we have chosen ResNet-50 as feature extractor, although others can also be used, such as VGG or DenseNet. In Section 5, a comparison between these three extractors is made to analyze the goodness of EvoPruneDeepTL with them.

These features are used as the input for the fully-connected layers. We introduce two different compositions of these fully-connected layers:

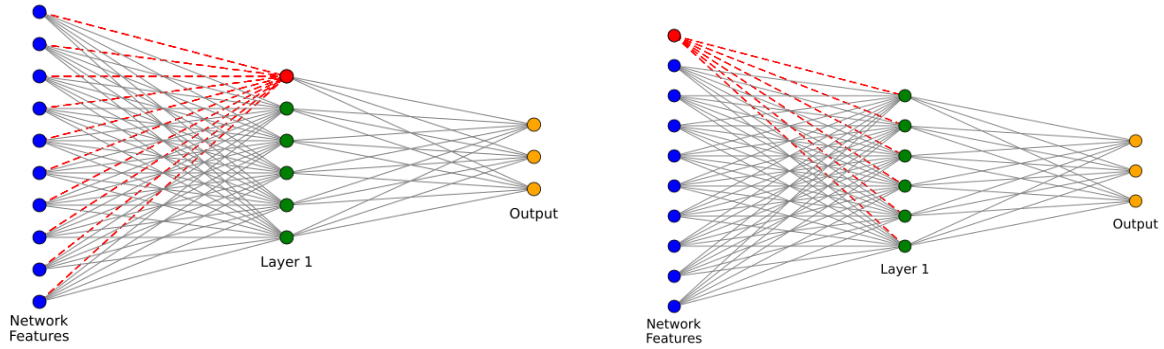
- Single fully-connected layer: it is composed of a single layer with 512 neurons, followed by the output layer.
- Two fully-connected layers: this architecture has two layers of 512 each, and the output layer connecting the output of the last fully connected layer to as many neurons as the number of classes to be discriminated in the dataset.

Moreover, activations of each fully connected layer are selected to be Rectifier Linear Unit (ReLU) functions. The output layer resorts to a SoftMax activation, which renders a probability distribution over the classes that compose the task at hand. The output neuron with the highest SoftMax probability leads to the predicted class of the input image.

EvoPruneDeepTL stands for the usage of sparse layers. By definition, a sparse layer has few active connections. A key object in this environment is the adjacency matrix. This matrix is key in our study

because it is used to create a sparse layer from it. It allows decoding an individual evolved via the GA to yield, as a result, a neural network with a sparse layer. Taking a look about this matrix, EvoPruneDeepTL performs the pruning in relation to the connections/neurons that compose the input of the neuron. Consequently, there may be some neurons of the second layer which have no connection from the previous layer.

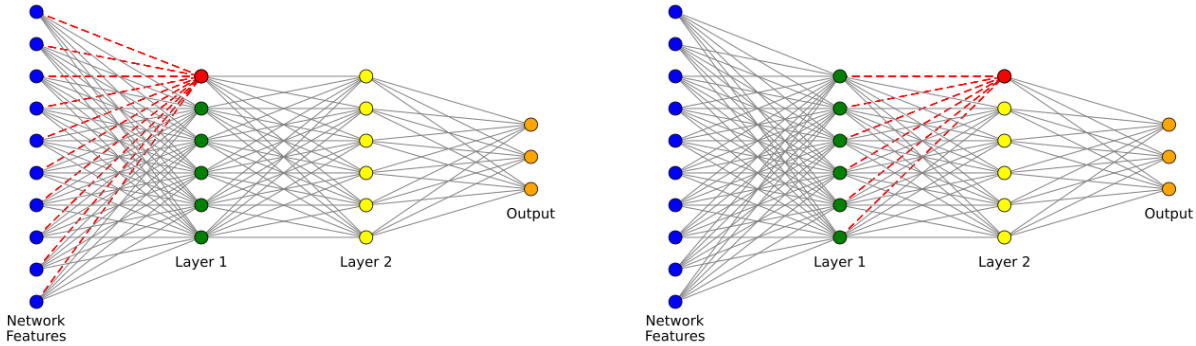
Based on the two network architectures described above, we consider different scenarios where EvoPruneDeepTL can be applied. We present these scenarios in the following figures, in which the red-dashed lines indicate the effects of the pruning. In addition to that, we have grouped the models in terms of the number of the last layers. The first model is the application of the EvoPruneDeepTL to prune model with one layer, which is shown in Fig. 4a. Moreover, when the pruning is made with networks with two layers, three cases come up: pruning the first layer (see Fig. 5a), pruning the second layer (see Fig. 5b for these cases), or both at the same time, which is the combination of the last two cases. Lastly, EvoPruneDeepTL is also able to prune the characteristics that are extracted from the network. This approximation is called Feature Selection because EvoPruneDeepTL prunes the features that are less important to enhance the accuracy of the network. Fig. 4b illustrates how pruning in this last scenario reduces to a selection of features.



(a) Pruning one layer

(b) Pruning extracted features

Figure 4: Visualization of pruning architectures with one layer



(a) Pruning the first layer

(b) Pruning the second layer

Figure 5: Visualization of pruning architectures with two layers

4. Experimental Framework

In this section, we describe the experimental framework followed in our study. First, we give a brief description of the analyzed datasets. Then, the training setup is presented, emphasizing the parameters of

EvoPruneDeepTL and the experimental conditions.

4.1. Datasets

In our study, we have chosen several diverse and representative datasets that are suitable for TL due to their size, as they require less training and inference time. Therefore, these datasets are suitable for population metaheuristics, as many individuals are evaluated. The selected data sets are shown in Table 1, which portrays their main characteristics for our experiments.

Table 1: Datasets used in the experiments.

Dataset	**Image Size**	**# classes**	**# Instances (train / test)**
SRSMAS	$(299, 299)$	14	333 / 76
RPS	$(300, 300)$	3	2520 / 372
LEAVES	$(256, 256)$	4	476 / 120
PAINTING	$(256, 256)$	5	7721 / 856
CATARACT	$(256, 256)$	4	480 / 121
PLANTS	$(100, 100)$	27	2340 / 236

These datasets are diverse and taken from the literature:

- SRSMAS ([dataset] Gómez-Ríos et al., 2019) is a dataset to classify coral reef types with different classes and high distinction difficulty.
- RPS ([dataset]Laurence Moroney, 2019) is a dataset to identify the gesture of the hands in the popular Rock Paper Scissors game from images that have different positions and different skin colors.
- LEAVES is composed of images of healthy and unhealthy citrus leaves, with different shades of green ([dataset]Hafiz Tayyab Rauf et al., 2019).
- PLANTS is another dataset from the natural environment ([dataset] Singh et al., 2020, May), in which the task is to differentiate between leaves of different plants such as tomato, apple or corn, among others.
- CATARACT comes from the medical domain ([dataset]Sungjoon Choi, 2020), whose purpose is to classify different types of eye diseases.
- PAINTING is related to the painting world ([dataset]Virtual Russian Museum, 2018). The images in this dataset have been taken from a museum and the task is to recognize different types of paintings.

Examples for several of the above datasets are shown in Fig. 6.

4.2. Training setup

The evaluation of EvoPruneDeepTL requires splitting the images of the datasets in train and test subsets. As the results could strongly depend on the train and test sets, we have applied in SRSMAS and LEAVES a 5-fold cross-validation [2]

For the remaining datasets, the train and test had already been defined beforehand, so we have used them for the sake of replicability.

The training is done using SGD as optimizer, a batch size of 32 images, and maximum 600 epochs, but the training stops when no improvement of loss is obtained in ten consecutive epochs. The model with the

Figure 6: Images of datasets. Top: SRSMAS examples. Middle: RPS examples. Bottom: LEAVES examples.

Table 2: Parameters of EvoPruneDeepTL.

Parameter	**Value**
Maximum Evals	200 (one layer) 300 (both layer)
# Runs	5
Population size	30
NAM	3
p_{mut}	0.07
Batch Size	32

greater accuracy on the training set is saved. As we apply TL, only the last layers are trained, whereas the remaining ones are frozen with the parameter values imported from the pre-trained ResNet-50 network.

The parameters of EvoPruneDeepTL are indicated in Table 2. We have set the maximum evaluations to two different values, 200 and 300, because there are some experiments we carry out to analyze the behavior of EvoPruneDeepTL that need an adaptation of this value because the search space in these experiments is wider. The size of the population of networks that our model evolves at each generation is set to 30, the mutation probability is p_{mut} and the NAM operator chooses the second parent among 3 candidates. The best solution found in terms of accuracy is returned. We note that in case of several solutions with the same accuracy, the returned solution is the configuration with the lowest percentage of active neurons. Note that the number of runs and total function evaluations is kept low to meet a computationally affordable balance between performance and the high execution times required for simulation. This is shown in Table 3, in which the average time per execution of the models with two layers is indicated. Unfortunately, this limited number of runs per experiment impedes the application of statistical tests to assess the significance of the reported differences, as tests conventionally used for this purpose require larger sample sizes to reach meaningful conclusions.

[2]Sets for 5 fold CV for SRSMAS and LEAVES:
https://drive.google.com/drive/folders/1Xf7OeZyWDDG-_Y4VX_nnAdfz3Kwhy8LU?usp=sharing. Last Access: 28/01/2022

Table 3: Average time per run of EvoPruneDeepTL.

Dataset	First Layer	Second Layer	Both Layers
SRSMAS	9h 41min	10h 45min	15h 27min
RPS	4h 23min	5h 13min	8h 00min
LEAVES	10h 26min	16h 01min	17h 45min
PAINTING	13h 1min	15h 24min	22h 45min
CATARACT	2h 3min	2h 22min	3h 26min
PLANTS	6h 3min	6h 11min	10h 00min

All the following experiments have been carried out using Python 3.6 and a Keras/Tensorflow implementation deployed and running on a Tesla V100-SXM2 GPU. The code is published in a open repository in GitHub.[3]

5. Results and Discussion

In this section, we analyze the behavior of EvoPruneDeepTL. In order to show the benefits of using EvoPruneDeepTL, we propose four research questions (RQ) that they are going to be answered with different and diverse experiments over several datasets are carried out. We will show tables with the results of these experiments and we will analyze them to ensure the benefits of EvoPruneDeepTL. These RQ are the following ones:

(**RQ1**) Which is the performance of EvoPruneDeepTL against fully-connected models?

We compare EvoPruneDeepTL against non-pruned models comprising fully-connected layers to study which model obtains a better performance in the experiments. Moreover, we remark the flexibility of EvoPruneDeepTL applying it with one and several layers.

(**RQ2**) Which would be better, to remove neurons or connections?

We compare EvoPruneDeepTL using the two alternatives explained in the previous section: 1) pruning the neurons or 2) each individual represents exact connections between neurons, allowing for a more finely grained evolution of the connections. The goal of this section is to check which representation obtains the best results. On the one hand, the neuron representation of the length of chromosomes is shorter, so the domain search is smaller. On the other hand, the connections representation is a more fine-detail representation, so it could potentially allow the algorithm to obtain better results.

(**RQ3**) Which is the performance of EvoPruneDeepTL when compared to efficient pruning methods?

We compare the performance of EvoPruneDeepTL against several efficient pruning methods published in the literature for compressing CNN networks: Polynomial Decay (Zhu & Gupta, 2018, April), Weight Pruning (Han et al., 2015, Dececember) and Neuron Pruning (Srinivas & Babu, 2015, September). This comparison of EvoPruneDeepTL and the CNN models is made in terms of accuracy and model compression.

(**RQ4**) Which would be better, pruning fully-connected layers or performing Feature Selection?

A particular case of EvoPruneDeepTL stands when the optimization of the network is done with one fully-connected layer and features are evolved towards the fittest for the problem at hand. Our

[3] EvoPruneDeepTL repository: https://github.com/ari-dasci/S-EvoDeepTLPruning.

aim is to check if this scheme improves the overall performance of the model over a dataset in terms of the accuracy of the models.

(**RQ5**) How does EvoPruneDeepTL perform when applied to different pre-trained networks?

We compare EvoPruneDeepTL with different feature extractors. From RQ1 to RQ4, ResNet-50 is used to extract the features or characteristics for the considered datasets. Experiments devised for this RQ aim to examine whether EvoPruneDeepTL adapts suitably to other feature extractors such as DenseNet-121 and VGG-19, so that better-performing pruned networks are produced by our proposal also for these feature extractors.

(**RQ6**) Can EvoPruneDeepTL adapt efficiently their pruned knowledge to changes in the modeling task, showing robustness?

We analyze the behavior of EvoPruneDeepTL when the datasets change. In this case, we have selected some datasets from our study, and we have done several modifications, removing partially or totally a class. Within these changes, we want to show both the robustness of the EvoPruneDeepTL in different situations and that the pruning models obtained by the GA of EvoPruneDeepTL have been adapted to each one of these situations.

This section is divided in Section 5.1, where the comparison of the diverse representations of pruning that EvoPruneDeepTL makes against the reference models is presented to answer RQ1. Next, Section 5.2 discusses whether EvoPruneDeepTL should operate over neurons or connections to analyze RQ2. Section 5.3 provides a complete comparison among EvoPruneDeepTL and other efficient pruning methods in order to solve RQ3. Section 5.4 explains the approximation of Feature Selection. A whole comparison against all the previous models is made to assess the importance of the Feature Selection to answer RQ4. Section 5.5 shows the comparison of the best two models of EvoPruneDeepTL with different feature extractors. Lastly, in Section 5.6, EvoPruneDeepTL is challenged, with several modifications of the used datasets, to grasp the relevant features of these datasets and to analyze the robustness of our proposal.

5.1. Answering RQ1: Pruning

In this section, we assess the performance gaps between the proposed EvoPruneDeepTL against other reference models to answer RQ1. In each subsection, several and diverse experiments are carried out to present results that assure the quality of EvoPruneDeepTL when it is compared to other models. This pruning section is composed of Section 5.1.1, in which we compare EvoPruneDeepTL against reference models with only one layer; of Section 5.1.2 we make the same experiments but with two layers and the evolution of one of them, and of Section 5.1.3, that shows the evolution of two consecutive layers at the same time.

In the following, we describe the different reference models:

- The first reference is composed of fully-connected layers of 512 units and the output layer. That is equivalent to the model with all neurons in active mode (all gens to 1). This model is the one with all active neurons, we call it *Not Pruned*.
- A grid search scheme is compared to EvoPruneDeepTL to check whether the improvement made by EvoPruneDeepTL could be obtained with a simple search over the percentage of neurons of the fully-connected layer. We have tested the fully-connected model with different number of neurons: 10% to 90% of its total units increasing this percentage by 10% (including both), and for each dataset we have identified the number of neurons which gives the best accuracy.
- The best result of the above models is also noted and it is called *Best Fixed*. When implemented over both layers, pruning is referred to as *Best Fixed Both*.

5.1.1. Pruning neurons of one fully-connected layer

This section introduces the results of pruning models with only one fully-connected layer. Table 4 shows the comparison of EvoPruneDeepTL against the reference models. In this case, the reference model only has one fully-connected layer composed by 512 units and the output layer.

For each dataset, the first row shows the obtained average accuracy by the models over the test set, whereas the second row informs about the average percentage of active neurons.

Table 4: Average results of EvoPruneDeepTL against not pruned models with one fully-connected layer.

Dataset	Measure	Not Pruned	Best Fixed	EvoPrune DeepTL
SRSMAS	Accuracy	0.832	0.866	**0.885**
	% Active neur.	100	20	25
RPS	Accuracy	0.938	0.938	**0.954**
	% Active neur.	100	40	46
LEAVES	Accuracy	0.923	0.927	**0.935**
	% Active neur.	100	80	38
PAINTING	Accuracy	0.939	0.945	**0.951**
	% Active neur.	100	60	46
CATARACT	Accuracy	0.703	0.719	**0.732**
	% Active neur.	100	70	39
PLANTS	Accuracy	0.432	0.432	**0.480**
	% Active neur.	100	10	49

These results show how EvoPruneDeepTL is capable of distinguishing the pruning configurations that lead towards an improvement of performance of the models, as it obtains a greater accuracy in all the datasets for every reference model. Moreover, in most datasets, a higher compression ratio than the best fully-connected model is also achieved.

5.1.2. Pruning neurons of two fully-connected layers

In this section, our challenge is to improve the performance of a two fully-connected layer network. For that reason, EvoPruneDeepTL is applied to each layer individually.

The results of applying EvoPruneDeepTL to each layer individually are shown in Table 5, where **First Layer** indicates the case of the evolution of the first layer, and **Second Layer** describes the other case. In this case, Both *Not Pruned* and *Best Fixed* are the reference models with two fully-connected layer.

In this case, results follow the same path as the previous one: in all the datasets, EvoPruneDeepTL achieves an improvement of the accuracy over the reference models. Moreover, the Second Layer case obtains more compressed networks than the First Layer option.

Comparing the results of the scheme of one and two layers, both have similar results, only in RPS and CATARACT the difference in terms of accuracy is higher. Thus, these experiments have shown the ability of EvoPruneDeepTL of improving the overall performance of networks and, at the same time, reducing their complexity.

5.1.3. Pruning neurons of both layers

In the previous sections, we have tested EvoPruneDeepTL to solve problems via the evolution of a single layer. In this section we increase the difficulty of the problem: the evolution of two consecutive fully-connected layers.

From the previous experiments, we have run EvoPruneDeepTL with 200 evaluations, but we have noticed that this number of evaluations might not be enough. This is due to the fact that we have now individuals

Table 5: Average results of EvoPruneDeepTL against not pruned models with two fully-connected layers.

Dataset	Measure	First Layer			Second Layer		
		Not Pruned	Best Fixed	EvoPrune DeepTL	Not Pruned	Best Fixed	EvoPrune DeepTL
SRSMAS	Accuracy	0.858	0.858	**0.883**	0.858	0.860	**0.884**
	% Active neur.	100	100	46	100	80	47
RPS	Accuracy	0.922	0.938	**0.959**	0.922	0.949	**0.969**
	% Active neur.	100	30	37	100	30	16
LEAVES	Accuracy	0.919	0.926	**0.937**	0.919	0.929	**0.935**
	% Active neur.	100	40	28	100	60	12
PAINTING	Accuracy	0.939	0.944	**0.950**	0.939	0.941	**0.951**
	% Active neur.	100	60	53	100	90	53
CATARACT	Accuracy	0.703	0.711	**0.740**	0.703	0.703	**0.735**
	% Active neur.	100	70	63	100	100	59
PLANTS	Accuracy	0.402	0.448	**0.479**	0.402	0.441	**0.483**
	% Active neur.	100	10	45	100	50	37

of size 1024, 512 for each layer, and the search space is larger than in the rest of experiments. We have therefore also carried out the experiments with 300 function evaluations.

In Table 6, we show the results for reference models and EvoPruneDeepTL with 300 evaluations. The reference models stand the same as in the previous cases, but as they are implemented over both layers, the pruning is now referred to as *Best Fixed Both*.

In some cases, the percentage of remaining active neurons is higher than in the first and second layer models, but that is due to the complexity of this new problem. However, the performance of the network in these experiments indicates that the best option for pruning is achieved when the evolution is done to two consecutive layers.

Table 6: Average results of EvoPruneDeepTL against not pruned methods evolving two consecutive layers.

Dataset	Measure	Not Pruned	Best Fixed Both	EvoPrune DeepTL
SRSMAS	Accuracy	0.858	0.863	**0.885**
	% Active neur.	100	50	64
RPS	Accuracy	0.922	0.946	**0.978**
	% Active neur.	100	90	12
LEAVES	Accuracy	0.919	0.934	**0.936**
	% Active neur.	100	15	34
PAINTING	Accuracy	0.939	0.949	**0.953**
	% Active neur.	100	40	51
CATARACT	Accuracy	0.703	0.735	**0.746**
	% Active neur.	100	85	63
PLANTS	Accuracy	0.402	0.466	**0.491**
	% Active neur.	100	55	41

These results prove the attainment of a sequential process to make pruning of DL models by adding layers and then, evolving their neurons to achieve a reduced configuration of the network. This process rises the performance of the models in terms of accuracy.

5.2. Answering RQ2: which would be better, to remove neurons or connections?

This section is devised to formally answer the RQ2, which is to decide if it is better to perform pruning of whole neurons or either single connections, by comparing different EvoPruneDeepTL chromosome representations: neurons and connections, as we described in Section 3. Two representations are shown in this section: the neuron representation, in which a gen represents the connections of a neuron, and the connections representation, in which a gene represents a specific connection in the sparse layer. Neuron representation obtains shorter chromosomes than the connections one. Meanwhile, the connection representation leads to a more detailed representation and a larger domain search.

Table 7: Average results of EvoPruneDeepTL against edges models.

		One Layer		**Two Layers**		
Dataset	Measure	Edges	EvoPruneDeepTL	Edges	EvoPruneDeepTL Layer 1	EvoPruneDeepTL Layer 2
SRSMAS	Accuracy	0.875	**0.885**	0.875	0.883	**0.884**
	% Active neur.	43	25	46	46	47
RPS	Accuracy	0.952	**0.954**	0.952	0.959	**0.969**
	% Active neur.	29	46	37	37	16
LEAVES	Accuracy	0.932	**0.935**	0.933	**0.937**	0.935
	% Active neur.	45	38	45	28	12
PAINTING	Accuracy	0.949	**0.951**	0.950	0.950	**0.951**
	% Active neur.	48	46	53	48	53
CATARACT	Accuracy	0.729	**0.732**	0.737	**0.740**	0.735
	% Active neur.	69	49	66	63	59
PLANTS	Accuracy	0.457	**0.480**	0.463	0.479	**0.483**
	% Active neur.	64	49	45	45	37

Table 7 shows for each dataset and representation the mean accuracy and % of active connections for both pruning methods. The connection strategy is named *Edges*. The results show that even though there are some cases in which the edges evolution achieves a similar performance of the network, the evolution of the neurons presents more robust results. The models working at the neuron level are even able to further reduce the number of active neurons in some datasets.

As a conclusion of this experiment, we can confirm that using the neuron approach is the best representation and that the second layer model gives us more consistent results than the first layer pruning model, both in accuracy and in reduction of the model.

5.3. Answering RQ3: Comparing EvoPruneDeepTL with efficient methods for CNN pruning

This section is devised to analyze the RQ3 comparing EvoPruneDeepTL to other well known network pruning methods to present results that measure the performance of our model against these methods. This comparison is conducted in terms of quality and computational complexity, aimed to prove the potential of EvoPruneDeepTL with respect to other pruning counterparts. To this end, we implement two different pruning methods, namely, weight pruning and neuron pruning. These methods have a parameter in common, $S_f \in \mathbb{R}(0, 1)$, which denotes the target pruning percentage. It is set to the same percentage of reduction that EvoPruneDeepTL has obtained in the experiments discussed previously. Next, we briefly describe each of such methods:

- `weight` (Han et al., 2015, Dececember): Parameters with lower values are pruned at once. This method operates over the whole parameter set in the layer to be optimized. (Parameters: S_f)
- `polynomial decay` (Zhu & Gupta, 2018, April): Parameters are pruned guided by a Polynomial Decay schedule to the specified sparsity value. Between pruning steps, the network is allowed to fine tune for 5 epochs. This model is also applied over the whole parameter set in the layer to be optimized. Parameters used in the experimentation are listed in Table 8.
- `neuron` (Srinivas & Babu, 2015, September): Neurons with lower mean input connection values are pruned. (Parameters: S_f) as in Figs. 4 and 5. (Parameters: S_f)

Table 8 summarizes the value of the parameters of Polynomial Decay algorithm, which have been adapted to our experiments. Then, given a desired sparsity value of S, the sparsity is updated over a span of k pruning steps following the next equation

$$S_k = S_f + (S_i - S_f) \cdot \left(1 - \frac{K_k - K_i}{K_f - K_i}\right)^{\alpha} \text{ if } K_k \bmod F = 0 \tag{3}$$

wherein parameters are described as follows:

- $S_{i,f} \in \mathbb{R}(0, 1)$ are the initial and final sparsity percentages.
- S_f depends on the experiment. It is the percentage of pruning that EvoPruneDeepTL has achieved and the end of the generations.
- $K_{i,f} \in \mathbb{N}$ configures at what training step the pruning algorithm starts and ends.
- $K_k \in \mathbb{N}(K_i, K_f)$ is the current step.
- nb is the number of batches. It is calculated as the length of the training set divided by the batch size.
- F configures the frequency at which Equation 3 is computed.

Table 8: Parameter values of Polynomial Decay.

Parameter	**Value**
S_i	0.1
K_i	0
K_f	$nb \cdot 25$
F	$nb \cdot 5$
α	3.0

Parameters of the Polynomial Decay model are chosen to achieve a tradeoff between network recovery and the number of training epochs. Given the nature of this model, Polynomial Decay implies more training epochs than the implemented neuron and weight pruning methods. This fact could make the comparison between such methods unfair if the additional training epochs introduced by the Polynomial Decay model are high compared to the initial training epochs (i.e. 600). To avoid this situation, Polynomial Decay is configured so that it sufficiently guarantees network recovery for all datasets while a minimal amount of extra training epochs are carried out, just an extra 4% from the initial 600 epochs (i.e. 25 extra epochs).

Our analysis aims to verify whether the performance of the above efficient pruning methods are comparable to EvoPruneDeepTL in terms of solution quality (accuracy) when they are configured to prune the same amount of parameters. Thus, the experimentation is carried out for the previously four cases discussed, selecting the average outcomes from the experimentation conducted in this point.

First, we show the results of this comparison when only a fully-connected layer is evolved. Table 9 shows the results for this case. EvoPruneDeepTL outperforms the CNN models in five out of the six cases, but only in PAINTING these results are better for the Polynomial Decay or Weight models.

Table 9: Average results of EvoPruneDeepTL against efficient CNN for one layer models.

		One Layer			
Dataset	Measure	Weight	Poly. Decay	Neuron	EvoPrune DeepTL
SRSMAS	Accuracy	0.805	0.823	0.745	**0.885**
	% Active neur.	25	25	25	25
RPS	Accuracy	0.917	0.927	0.869	**0.954**
	% Active neur.	46	46	46	46
LEAVES	Accuracy	0.918	0.920	0.886	**0.935**
	% Active neur.	38	38	38	38
PAINTING	Accuracy	0.993	**0.994**	0.874	0.951
	% Active neur.	46	46	46	46
CATARACT	Accuracy	0.678	0.679	0.658	**0.732**
	% Active neur.	39	39	39	39
PLANTS	Accuracy	0.406	0.411	0.365	**0.480**
	% Active neur.	49	49	49	49

Table 10: Average results of EvoPruneDeepTL against efficient CNN pruning methods for two layers models.

		First Layer				Second Layer				Both Layers			
Dataset	Measure	Weight	Poly. Decay	Neuron	EvoPrune DeepTL	Weight	Poly. Decay	Neuron	EvoPrune DeepTL	Weight	Poly. Decay	Neuron	EvoPrune DeepTL
SRSMAS	Accuracy	0.795	0.815	0.775	**0.883**	0.834	0.837	0.779	**0.884**	0.845	0.847	0.647	**0.885**
	% Active neur.	46	46	46	46	47	47	47	47	64	64	64	64
RPS	Accuracy	0.886	0.911	0.803	**0.959**	0.845	0.911	0.696	**0.969**	0.694	0.899	0.490	**0.978**
	% Active neur.	37	37	37	37	16	16	16	16	12	12	12	12
LEAVES	Accuracy	0.913	0.918	0.812	**0.937**	0.904	0.919	0.712	**0.935**	0.911	0.925	0.747	**0.936**
	% Active neur.	28	28	28	28	12	12	12	12	34	34	34	34
PAINTING	Accuracy	**0.995**	0.993	0.850	0.950	0.937	0.938	0.920	**0.951**	0.934	0.940	0.853	**0.953**
	% Active neur.	53	53	53	53	53	53	53	53	51	51	51	51
CATARACT	Accuracy	0.668	0.684	0.673	**0.740**	0.694	0.689	0.648	**0.737**	0.686	0.696	0.611	**0.746**
	% Active neur.	63	63	63	63	59	59	59	59	63	63	63	63
PLANTS	Accuracy	0.408	0.403	0.343	**0.479**	0.392	0.420	0.313	**0.482**	0.393	0.411	0.278	**0.491**
	% Active neur.	45	45	45	45	37	37	37	37	41	41	41	41

Second, Table 10 shows the results for the evolution of models with two layers, where **First Layer** indicates the cases of the evolution of the first layer and **Second Layer** describes the cases of the second layer. Results point out that EvoPruneDeepTL outperforms most of the methods in all the models and datasets. This case presents similar results as the one layer case because only in the PAINTING dataset EvoPruneDeepTL has a lower performance in relation to the literature methods. As a result of that, EvoPruneDeepTL's robustness in performance over the literature methods has been shown in one-layer and two-layer networks.

Lastly, we compare the execution times for all the models. Evolutionary approaches are known to converge slowly in highly-dimensional search spaces, as the one tackled in this paper. For that reason, in this section, we also want to compare the required time of EvoPruneDeepTL and the other traditional approaches. Table 11 shows the time in seconds for each model. From these results, in terms of computational efficiency, our method suffers from the convergence slowness derived from the exploration of large search spaces.

To summarize, in this section we have fairly compared EvoPruneDeepTL to other well-known pruning methods, such as weight pruning and neuron pruning, guided by different pruning techniques. Evo-

PruneDeepTL is distinguished from other pruning methods due to the fact that they are advocate for shrinking the through their pruning process, but with an admissible decrease of the accuracy. Although our model is slower in terms of execution time, it scores higher accuracy levels than those of traditional pruning counterparts. Therefore, we conclude that EvoPruneDeepTL excels at determining which parameters to tune in neural networks with imported knowledge from other related tasks.

Table 11: Times in seconds per run of EvoPruneDeepTL against efficient CNN pruning methods with one and two layers models.

	One Layer				**Two Layers**					
Dataset	Weight	Poly. Decay	Neuron	EvoPruneDeepTL	Weight	Poly. Decay	Neuron	EvoPruneDeepTL Layer 1	EvoPruneDeepTL Layer 2	EvoPruneDeepTL Both Layers
SRSMAS	1,995	2125	1,995	34,510	2,395	2,545	2,398	34,856	38,731	55,596
RPS	1,674	1,893	1,674	19,851	1,229	1,379	1,229	15,758	18,790	28,774
LEAVES	2,425	2,560	2,425	35,243	2,430	2,565	2,430	37,561	57,695	63,897
PAINTING	1,386	1,508	1,386	61,734	2,903	3,243	2,903	46,856	55,414	81,913
CATARACT	594	627	584	6,768	449	473	449	7,392	8,529	12,350
PLANTS	298	407	298	28,456	270	370	270	21,788	22,235	35,998

5.4. Answering RQ4: Feature Selection

The RQ4 establishes the dichotomy of choosing pruning or feature selection for the given problem. For that reason, in this section we analyze the FS model by conducting the same group of experiments of the previous sections, to compare it against EvoPruneDeepTL to decide which one scores best among them. The FS scheme is a particular case of EvoPruneDeepTL if only one fully-connected layer composes the configuration of the network and the pruning and GA are focused on the extracted features of the ResNet-50 model.

Table 12 shows the results for this model against the reference methods. This case follows the same similarities of the previous ones, as FS obtains the best average results for all the datasets.

Table 12: Average results for Fetaure Selection against non pruning methods.

Dataset	Measure	Not Pruned	Best Fixed	Feature Selection
SRSMAS	Accuracy	0.832	0.866	**0.884**
	% Active neur.	100	20	60
RPS	Accuracy	0.938	0.938	**0.985**
	% Active neur.	100	40	45
LEAVES	Accuracy	0.923	0.927	**0.943**
	% Active neur.	100	80	59
PAINTING	Accuracy	0.939	0.945	**0.958**
	% Active neur.	100	60	55
CATARACT	Accuracy	0.703	0.719	**0.747**
	% Active neur.	100	70	55
PLANTS	Accuracy	0.432	0.432	**0.472**
	% Active neur.	100	10	68

Similarly to the previous sections, we have also compared this model with the CNN pruning methods with only one layer, as shown in Table 13. In this case, in four out of six datasets the Feature Selection outperforms these methods, but in LEAVES and PAINTING Weight and Polynomial Decay perform better than our model.

Table 13: Average results of Feature Selection against efficient CNN pruning methods.

		Feature Selection			
Dataset	Measure	Weight	Poly. Decay	Neuron	Feature Selection
SRSMAS	Accuracy	0.841	0.878	0.802	**0.884**
	% Active neur.	60	60	60	60
RPS	Accuracy	0.913	0.926	0.869	**0.985**
	% Active neur.	45	45	45	45
LEAVES	Accuracy	**0.947**	0.940	0.946	0.943
	% Active neur.	59	59	59	59
PAINTING	Accuracy	0.962	**0.968**	0.883	0.958
	% Active neur.	55	55	55	55
CATARACT	Accuracy	0.696	0.689	0.687	**0.747**
	% Active neur.	55	55	55	55
PLANTS	Accuracy	0.421	0.317	0.402	**0.472**
	% Active neur.	68	68	68	68

In this section, we have compared our FS scheme against reference models and efficient pruning methods published in the literature. The results shed light over the benefits of this model as it is also able to achieve a great performance over the reference models and also, in most cases, against the CNN pruning methods.

The global results for EvoPruneDeepTL and its different versions are presented in Table 14. The rows show the achieved accuracy and the percentage of improvement in relation to the best reference models for each model.

Table 14: Results and percentage of improvement for each version of EvoPruneDeepTL in relation to each reference model.

Dataset	Measure	Pruning Model One Layer	Pruning Model Both Layers	Feature Selection
SRSMAS	Accuracy	**0.885**	**0.885**	0.884
	% Improvement	1.9	2.2	1.8
RPS	Accuracy	0.954	0.978	**0.985**
	% Improvement	1.6	3.2	4.7
LEAVES	Accuracy	0.935	0.936	**0.943**
	% Improvement	0.8	0.2	1.6
PAINTING	Accuracy	0.951	0.953	**0.958**
	% Improvement	0.6	0.4	1.3
CATARACT	Accuracy	0.732	0.746	**0.747**
	% Improvement	1.3	1.1	2.8
PLANTS	Accuracy	0.480	**0.491**	0.472
	% Improvement	4.8	2.5	4.0

Reviewing the results of EvoPruneDeepTL and FS, we confirm that FS is the best model , as it obtains the best accuracy levels in four out of six datasets. Furthermore, the pruning of both layers carried out by EvoPruneDeepTL also attains very notable performance levels.

Moreover, if we consider the evolution using the pruning model, the evolution of both layers yields the best results in terms of mean accuracy for each dataset. However, when comparing pruning and FS, the latter has more robust models: it achieves the best performance in four datasets, and it is also shown in the improvement percentage for each dataset.

In conclusion, it is shown with empirical evidence that pruning can be done by evolving the fully-connected layers, specifically, by evolving their neurons to get the fittest configuration that reports an improvement of the network performance. An evolutionary feature selection based on the extracted features also achieves a great network performance, both in improving the accuracy and in reducing its complexity.

5.5. Answering RQ5: Comparing different EvoPruneDeepTL with different feature extractors

This section is devised to formally compare EvoPruneDeepTL with different networks that serve as feature extractor for each dataset to analyze. CNNs have shown their capability to overcome different and diverse classification problems by learning visual features that best correlate with the target variable of the task at hand. Transferring this knowledge to other problems with a similar domain, which is what TL stands for, also helps to the capability of generalization of the model that is devised for the target task, specially when the volume of data for that task is low.

In previous sections, we have observed that the combination of EvoPruneDeepTL with ResNet-50 has improved both reference models and pruning methods of the literature. However, in this section, we explore two other feature extractors and assess whether such performance gaps prevail. To determine performance gains of EvoPruneDeepTL when using these alternative feature extractors, both are tested over the two best performing scenarios of EvoPruneDeepTL, namely pruning both layers and feature selection. Moreover, we also include in the comparison these feature extractors with pruning algorithms from the literature, following the same experimental procedure described in preceding sections.

The chosen feature extractors are DenseNet-121 and VGG-19. The experiments with these networks have been carried out in the same conditions that the previous ones have been done. Table 15 shows the comparison of these networks against the reference models based on fully-connected layers.

Table 15: Average results of EvoPruneDeepTL with different networks evolving two consecutive layers against non pruning methods.

		ResNet-50			**DenseNet-121**			**VGG19**		
Dataset	Measure	Not Pruned	Best Fixed Both	EvoPrune DeepTL	Not Pruned	Best Fixed Both	EvoPrune DeepTL	Not Pruned	Best Fixed Both	EvoPrune DeepTL
SRSMAS	Accuracy	0.858	0.863	**0.885**	0.861	0.881	**0.890**	0.837	0.853	**0.885**
	% Active neur.	100	50	64	100	50	72	100	85	55
RPS	Accuracy	0.922	0.946	**0.978**	0.704	0.723	**0.754**	0.814	0.879	**0.922**
	% Active neur.	100	90	12	100	70	43	100	40	50
LEAVES	Accuracy	0.919	0.934	**0.936**	0.896	0.904	**0.915**	0.903	0.911	**0.917**
	% Active neur.	100	15	34	100	60	39	100	50	68
PAINTING	Accuracy	0.939	0.949	**0.953**	0.940	0.943	**0.947**	0.923	0.938	**0.945**
	% Active neur.	100	40	51	100	40	67	100	70	34
CATARACT	Accuracy	0.703	0.735	**0.746**	0.694	0.727	**0.741**	0.661	0.727	**0.759**
	% Active neur.	100	85	63	100	55	51	100	45	42
PLANTS	Accuracy	0.402	0.466	**0.491**	0.411	0.428	**0.456**	0.292	0.364	**0.374**
	% Active neur.	100	55	41	100	55	78	100	50	64

The results from the previous table show the ability of EvoPruneDeepTL to adapt itself to several feature extractors. For both DenseNet-121 and VGG-19, it improves the reference models. Taking a deep look at the three networks, ResNet-50 is the best of them, as it has the best improvement over several datasets. Nonetheless, the straight conclusion which is derived from these experiments is that EvoPruneDeepTL is able to adapt to different feature extractors and datasets.

Once we have seen that EvoPruneDeepTL has achieved better results than the reference models, now we inspect the performance of different pruning methods from the literature. For that reason, we compare

the model which prunes two consecutive layers against its similar models from the pruning methods. Table 16 shows the comparison of the three networks against these methods. Note that, for this comparison, we have used the best to models of the CNN pruning methods: Weights and Polynomial Decay. This holds for the rest of this section.

Table 16: Average results of EvoPruneDeepTL against efficient CNN pruning methods for pruning consecutive layers.

		Both Layers - ResNet50			**Both Layers - DenseNet-121**			**Both Layers - VGG19**		
Dataset	Measure	Weight	Poly. Decay	EvoPrune DeepTL	Weight	Poly. Decay	EvoPrune DeepTL	Weight	Poly. Decay	EvoPrune DeepTL
SRSMAS	Accuracy	0.845	0.847	**0.885**	0.862	0.865	**0.890**	**0.933**	0.869	0.885
	% Active neur.	64	64	64	72	72	72	55	55	55
RPS	Accuracy	0.694	0.899	**0.978**	0.815	**0.817**	0.754	0.803	0.842	**0.922**
	% Active neur.	12	12	12	43	43	43	50	50	50
LEAVES	Accuracy	0.911	0.925	**0.936**	**1.000**	0.907	0.915	**1.000**	0.916	0.917
	% Active neur.	34	34	34	39	39	39	68	68	68
PAINTING	Accuracy	0.934	0.940	**0.953**	0.897	0.901	**0.947**	0.923	0.922	**0.945**
	% Active neur.	51	51	51	67	67	67	34	34	34
CATARACT	Accuracy	0.686	0.696	**0.746**	0.587	0.593	**0.741**	0.661	0.686	**0.759**
	% Active neur.	63	63	63	51	51	51	42	42	42
PLANTS	Accuracy	0.393	0.411	**0.491**	0.251	0.249	**0.456**	0.284	0.292	**0.374**
	% Active neur.	41	41	41	78	78	78	64	64	64

The results shown in Table 16 give rise to interesting insights. To begin with, the first three columns are related to ResNet-50, which have a great performance over these reference models, and we know it from the previous sections. However, DenseNet and VGG are totally new in this kind of experiments. The reality is that both of these networks improve the CNN pruning methods when they are applied to fully-connected layers in most cases. Only in a few experiments are better than obtained by our proposal.

A global vision of these experiments suggests that DenseNet and VGG, just like ResNet, contribute to the discovery of pruned neural networks that maximize accuracy and reduce the number of active neurons. Moreover, these results verify that EvoPruneDeepTL is able to achieve for different networks better results than reference and efficient CNN pruning methods when the pruning is made in two consecutive layers. Thus, we have shown the adaption ability of EvoPruneDeepTL for this case using several networks (ResNet, DenseNet, and VGG).

The following scenario is the feature selection model which derives from EvoPruneDeepTL. This scenario, which encourages the pruning of the features extracted by the pre-trained network, has yielded the best results so far. Next, Table 17 shows the comparison of these networks when feature selection is performed.

In the previous sections, we have shown that EvoPruneDeepTL is able to prune the extracted features derived from the network, and this model has reached the best results of EvoPruneDeepTL. Table 17 shows that the pruning of the features that they have been extracted using different networks (DenseNet and VGG) also increases the performance of the networks, which is shown in the accuracy of these networks over the datasets. There are some cases in which non pruning methods have less active neurons, but their accuracy is lower than the models of EvoPruneDeepTL. For that reason, the feature selection keeps being the best of EvoPruneDeepTL models, as it has the best results so far.

The next, and final, step is to check the performance of this feature selection model against efficient CNN pruning methods from the literature. In this section, we have checked that, for models which prune two consecutive layers, EvoPruneDeepTL performs better than the pruning methods. Consequently, now we focus on this comparison, but in terms of the models which prune the extracted features. Table 18 shows the results of this comparison for the different networks.

The results show, not only that ResNet-50 has a great performance (same results as previous sections), that both DenseNet and VGG outperform the pruning methods when applied to prune the features extracted from the networks. Both new networks show a better performance in all the datasets than the reference

Table 17: Average results for Feature Selection with different networks against non-pruning methods.

Dataset	Measure	ResNet-50			DenseNet-121			VGG19		
		Not Pruned	Best Fixed	Feature Selection	Not Pruned	Best Fixed	Feature Selection	Not Pruned	Best Fixed	Feature Selection
SRSMAS	Accuracy	0.832	0.866	**0.884**	0.858	0.881	**0.896**	0.753	0.766	**0.869**
	% Active neur.	100	20	60	100	60	68	100	40	87
RPS	Accuracy	0.938	0.938	**0.985**	0.720	0.720	**0.839**	0.887	0.890	**0.982**
	% Active neur.	100	40	45	100	100	48	100	60	53
LEAVES	Accuracy	0.923	0.927	**0.943**	0.896	0.902	**0.921**	0.852	0.876	**0.924**
	% Active neur.	100	80	59	100	10	60	100	10	68
PAINTING	Accuracy	0.939	0.945	**0.958**	0.934	0.941	**0.956**	0.924	0.924	**0.943**
	% Active neur.	100	60	55	100	90	63	100	100	77
CATARACT	Accuracy	0.703	0.719	**0.747**	0.669	0.702	**0.787**	0.628	0.661	**0.765**
	% Active neur.	100	70	55	100	30	57	100	10	66
PLANTS	Accuracy	0.432	0.432	**0.472**	0.394	0.415	**0.464**	0.335	0.352	**0.376**
	% Active neur.	100	10	68	100	90	67	100	40	66

Table 18: Average results of EvoPruneDeepTL with different networks against efficient CNN pruning methods for feature selection models.

Dataset	Measure	Feature Selection - ResNet50			Feature Selection - DenseNet-121			Feature Selection - VGG19		
		Weight	Poly. Decay	Feature Selection	Weight	Poly. Decay	Feature Selection	Weight	Poly. Decay	Feature Selection
SRSMAS	Accuracy	0.841	0.878	**0.884**	0.869	0.868	**0.896**	0.826	0.825	**0.869**
	% Active neur.	60	60	60	68	68	68	87	87	87
RPS	Accuracy	0.913	0.926	**0.985**	0.675	0.699	**0.839**	0.981	0.834	**0.982**
	% Active neur.	45	45	45	48	48	48	53	53	53
LEAVES	Accuracy	**0.947**	0.940	0.943	0.858	0.891	**0.921**	0.904	0.843	**0.924**
	% Active neur.	59	59	59	60	60	60	68	68	68
PAINTING	Accuracy	0.962	**0.968**	0.958	0.937	0.934	**0.956**	0.928	0.928	**0.943**
	% Active neur.	55	55	55	63	63	63	77	77	77
CATARACT	Accuracy	0.696	0.689	**0.747**	0.676	0.682	**0.787**	0.666	0.688	**0.765**
	% Active neur.	55	55	55	57	57	57	66	66	66
PLANTS	Accuracy	0.421	0.317	**0.472**	0.387	0.394	**0.464**	0.322	0.311	**0.376**
	% Active neur.	68	68	68	67	67	67	66	66	66

methods. For that reason, we conclude that the usage of EvoPruneDeepTL with these three networks has proven the capability to perform better than the pruning methods.

In this section, we have compared ResNet-50 with other two networks in two different scenarios: pruning consecutive layers and pruning the extracted features from the networks. The experiments show that EvoPruneDeepTL has proven its ability to adapt to other networks in both cases and has improved both reference models and pruning methods of the literature. For that reason, and in light of the results from the previous sections, we conclude that EvoPruneDeepTL has shown the ability of facing diverse tasks, as EvoPruneDeepTL has achieved a great performance when different networks are used either for pruning consecutive layer or pruning the features extracted from the network.

5.6. Answering RQ6: Analyzing the ability of EvoPruneDeepTL to adapt to relevant classes and robustness.

The purpose of this section is twofold. First, we want to analyze the goodness of EvoPruneDeepTL when modeling varying problems. In this case, we want to see how it adapts to the different classes that make up the datasets so that it captures the relevance of each of them. For a given dataset, we analyze each of its classes to determine if EvoPruneDeepTL is also able to have a good performance over it, and then, compare these results with the whole dataset.

The second objective of this section is the analysis of these results. Once EvoPruneDeepTL has modelled each of the classes for a dataset at hand, we check the quality of the obtained results. For that reason, we must check that results are not affected by the stochasticity induced by the usage of a genetic algorithm at the core of the proposed EvoPruneDeepTL. Recall that stochasticity implies that the output of the algorithm may not be the same, even with the same input. For that reason, our second objective is related to this factor, and we want to show that the effect of randomness has a low impact on EvoPruneDeepTL, i.e., the good results do not depend on the randomness.

In order to measure the effect of randomness in the pruned networks evolved by EvoPruneDeepTL, we resort to a similarity measure called Centered Kernel Alignment, CKA (Kornblith et al., 2019, June). CKA measures the similarity of trained neural networks, in compliance with several invariance properties that must be met for these particular computing structures (namely invariance to invertible linear transformation, invariance to orthogonal transformation and invariance to isotropic scaling). We compare the trained networks as a result of the application of EvoPruneDeepTL. This comparison answers the question about the robustness of EvoPruneDeepTL.

CKA is based on the Hilbert-Schmidt Independence Criterion, HSIC (Gretton et al., 2005, October). It compares two matrices ($\mathbf{K}$ and $\mathbf{L}$) and determines the level of independence between them, as it is shown in 4. In our case, these matrices are the structures that contain the weights of the trained neural networks. CKA takes a maximum value of 1 when the two inputs of CKA are the same matrix. The range of this measure is $[0, 1]$. This means that both matrix are very similar (in that case because they are identical). Thus, if the CKA value is high, then both matrix are similar.

$$CKA(K, L) = \frac{HSIC(K, L)}{\sqrt{HSIC(K, K)HSIC(L, L)}} \tag{4}$$

CKA is a measure which allows us make a double comparison. Note that this measure helps us to compare the genotype (chromosomes of the GA) against the phenotype (pruned networks of EvoPruneDeepTL). The interpretation of this measure follows that if the similarity of the chromosomes is high, then the CKA value should be also high (close to one, which is its maximum). Five independent executions have been made of EvoPruneDeepTL, so we have taken the output of each of them, and we have performed a double comparison based on this trained neural network, which is explained next:

1. Comparison against the closest element ($CKA_{Closest}$): we have the *Output* element as the best one for an execution. Then, we calculate the Hamming distance of the best element with respect to all the elements evaluated in the evolutionary process of EvoPruneDeepTL. The element with the smallest Hamming distance to the best one is denoted as *Closest*. Then, *Output* is compared against *Closest*. The robustness of EvoPruneDeepTL is tested in this comparison because high values of CKA when similar chromosomes are compared is essential, as it will show that the results are not due to randomness, but to the process that EvoPruneDeepTL performs.

2. Comparison against fully-connected, reference models (CKA_{Ref}): in this case, we compare *Output* against a fully-connected network with all the neurons activated. This comparison sheds light on the ability of EvoPruneDeepTL to learn which neurons are the best to solve the problem at hand. It also permits to explain the difference in terms of accuracy between the models that EvoPruneDeepTL develops against the reference models. This value of CKA quantifies the differences in accuracy between our models and the reference models, as EvoPruneDeepTL searches for the best neurons to remove the unnecessary ones, while reference models simply train the models without taking into account the neurons which should be removed.

In this section, we select the best two models of EvoPruneDeepTL for these experiments: pruning consecutive layers (both case) and pruning the features extracted from the networks (feature selection). Moreover, three datasets are considered in the experiments designed for this section: CATARACT, PAINTING and RPS. Lastly, we show different tables with the following structure: *DATASET-Class*. This means that the mentioned *DATASET* is analyzed without the class called *Class*.

For the CKA comparison, we will show two groups of tables, one per each type of model (pruning consecutive layer and feature selection). Moreover, each table is composed of the problem at hand and the mean values of the Hamming distance and CKA averaged over five executions of EvoPruneDeepTL. The rows of each table correspond to each *DATASET-Class* and the four columns represent in pairs the previously explained comparisons, $\text{CKA}_{Closest}$ and CKA_{Ref}, as we show the mean Hamming distance and its corresponding CKA mean value for each of the comparisons.

5.6.1. Analyzing the relevance of each class for a given dataset

Given a dataset of n classes, this approach performs a procedure that removes a whole class of the dataset and then, EvoPruneDeepTL is applied with that remaining data both in pruning consecutive layers and in feature selection. As a result of that process, n experiments are done for each dataset.

The structure of the these tables correspond with the usual structure of the rest of the paper, but now we show four columns. These columns represent, in pairs based on the model, the results of EvoPruneDeepTL versus the reference model without pruning. The difference between the pairs of columns is the model at hand: pruning consecutive layers or feature selection. We note that for the first two columns, the evolution process which lies in EvoPruneDeepTL is made with 300 evaluations and two-layers networks, meanwhile for the last columns, the process of pruning the extracted features is made with only one-layer networks and 200 evaluations, i.e., under the same conditions as the experiments in the previous sections.

The first results show the CATARACT dataset under these conditions. Table 19 shows these results. We conclude from these results that Cataract class is the easiest class in the dataset, as both models struggle with that class (third row of the table), but they improve their results with it. However, the results of EvoPruneDeepTL are better than the reference model. The number of active neurons at the end of the evolutionary process are reduced, in most cases, nearly by half. Moreover, the feature selection model is also able to decrease this number from its previous results (55% of remaining active neurons) in some cases. The same conclusion can be drawn in relation to pruning consecutive layers, as the full dataset has a mean percentage of active neurons of 63% and in three of fours cases this number is reduced. The results show that the pruning of the extracted features of the network, i.e., the feature selection which derives from EvoPruneDeepTL, is the best approximation for this dataset.

Table 19: Average results of CATARACT-Class with pruning consecutive layers and feature selection.

Dataset	Measure	EvoPruneDeepTL Both	No Pruning Both	EvoPruneDeepTL Feature Selection	No Pruning Feature Selection
CATARACT - Retina	Accuracy	**0.844**	0.801	**0.871**	0.697
	% Active neur.	43	100	54	100
CATARACT - Glaucoma	Accuracy	**0.846**	0.789	**0.857**	0.554
	% Active neur.	44	100	63	100
CATARACT - Cataract	Accuracy	**0.735**	0.732	**0.761**	0.614
	% Active neur.	76	100	49	100
CATARACT - Normal	Accuracy	**0.833**	0.805	**0.843**	0.655
	% Active neur.	53	100	57	100
CATARACT - Full	Accuracy	**0.746**	0.703	**0.747**	0.703
	% Active neur.	63	100	55	100

We focus now on the following dataset, RPS. Table 20 shows the results of these experiments. Results show that the *Paper* class makes an easier dataset, as all the approaches reach the maximum accuracy. The other two experiments show that EvoPruneDeepTL achieves a better performance to the reference models. For pruning consecutive layers, the number of remaining active neurons is higher in comparison with the full dataset, but in the feature selection model this number is very similar or even for RPS-Scissors is lower.

Table 20: Average results of general RPS pruning consecutive layers and feature selection.

Dataset	Measure	EvoPruneDeepTL Both	No Pruning Both	EvoPruneDeepTL Feature Selection	No Pruning Feature Selection
RPS - Paper	Accuracy	**1.000**	1.000	**1.000**	1.000
	% Active neur.	51	100	51	100
RPS - Rock	Accuracy	**0.985**	0.979	**1.000**	0.996
	% Active neur.	51	100	49	100
RPS - Scissors	Accuracy	**0.994**	0.955	**1.000**	0.955
	% Active neur.	33	100	23	100
RPS Full	Accuracy	0.978	0.922	0.985	0.938
	% Active neur.	12	100	45	100

The last considered dataset is PAINTING. Table 21 shows the results for each of the experiments which have been made for this dataset. The table shows that both of EvoPruneDeepTL models are constantly achieving better results than the reference models. However, the difference between the accuracy is higher in the feature selection models than the pruning consecutive layers. Taking into consideration the remaining active neurons, the feature selection has a similar degree of pruning in relation with the experiments which have been carried out for the full dataset experiments (55%). Similar conclusion can be drawn for the other model, as the mean percentage of active neuron for pruning consecutive layers is 51% and we have models with fewer active neurons, but also with higher percentage.

Table 21: Average results of general PAINTING pruning consecutive layers and feature selection.

Dataset	Measure	EvoPruneDeepTL Both	No Pruning Both	EvoPruneDeepTL Feature Selection	No Pruning Feature Selection
PAINTING - Sculpture	Accuracy	**0.942**	0.932	**0.947**	0.857
	% Active neur.	64	100	53	100
PAINTING - Painting	Accuracy	**0.959**	0.946	**0.961**	0.820
	% Active neur.	42	100	50	100
PAINTING - Iconography	Accuracy	**0.942**	0.920	**0.949**	0.835
	% Active neur.	30	100	57	100
PAINTING - Engraving	Accuracy	**0.979**	0.974	**0.979**	0.883
	% Active neur.	74	100	56	100
PAINTING - Drawings	Accuracy	**0.994**	0.989	**0.996**	0.944
	% Active neur.	57	100	57	100
PAINTING Full	Accuracy	0.953	0.939	0.958	0.939
	% Active neur.	51	100	55	100

These experiments shed light on a conclusion that it is not far from the previous sections. The feature selection model, which performs the pruning of the extracted features of the network, constitutes the best model for all the experiments, as it has the most difference between EvoPruneDeepTL and reference models. Nonetheless, the effect of pruning consecutive layers is also positive, as it is shown in the results of the previous tables.

The next part of this section is crucial to determine the robustness of EvoPruneDeepTL. Moreover, the differences in accuracy of the previous tables are going to be explained in the following tables. The key

element of the comparison is the CKA measure and the Hamming distance of the solutions. The combination of these values determine the key points that we have discussed at the beginning of this section.

Therefore, we are showing the CKA tables for each experiment to perform the commented double comparison in this section. These tables have a different structure from the last tables, so we explain how to interpret them. Both of them present a similar structure, but the table which shows the results for pruning consecutive layers has another column. This column shows the value of the CKA measure for the layer at hand. In the case of the prune of the extracted feature of the network (feature selection), as they only has one layer, this column is not required.

We show this pair of tables for each *DATASET-Class*. The first table shows the results of the feature selection model, and the second one presents the results of the pruning of consecutive layers.

The composition of the tables for feature selection models is the following one. After the first column which show the dataset at hand, the next two columns represent the comparison among the phenotype and genotype of EvoPruneDeepTL, i.e, chromosomes and pruned networks. Both mean Hamming distance of the five executions and $\text{CKA}_{Closest}$ are shown. The following group of two columns shows the other explained comparison of the reference models, which have all the neurons active. The metrics are the same as the previous case, but now the CKA (CKA_{Ref}) corresponds to the mean value from the best model to this reference model.

The composition of the table for the pruning of consecutive layers is similar to the previous case. However, another column is required for a more detailed explanation. This column gives information about the CKA value for the layer at hand. Due to the fact that we are comparing the whole chromosome, both Hamming distance values are common to both layers, but the CKA value is layer dependent. Then, in this table, we want to highlight that the best chromosome EvoPruneDeepTL obtains good values of the CKA measure for each of the layers of the model.

First, we show the CKA values for the CATARACT dataset in its four different cases of *DATASET-Class*. Table 22 and Table 23 show the results for feature selection and pruning consecutive layers, respectively. The results show for both models the robustness of EvoPruneDeepTL because the mean CKA of the best versus its closest chromosome in the history is a value extremely close to 1, which is the maximum value (this value is reached when the best solution is compared to itself).

Taking a more deep look at the results of the feature selection, we see that the Hamming distance is very low, which is a good result and also proves the robustness of EvoPruneDeepTL. The second group of columns, in which the comparison is made against a model with all neurons active, we see that the Hamming distance is higher and this has an impact on the CKA value, which is higher. The conclusion which derives from these experiments is that EvoPruneDeepTL learns to distinguish the valuable neurons which have an impact on the model. This CKA value is the explanation of the difference in accuracy in the previous experiments of this dataset between EvoPruneDeepTL models and reference models.

Table 22: Comparison of the CKA measure for feature selection in CATARACT-Class.

Dataset	**Feature Selection**			
	Hamming Distance EvoPruneDeepTL	$\text{CKA}_{Closest}$	Hamming Distance No Pruned Model	CKA_{Ref}
CATARACT - Retina	0.005	0.981	0.457	0.283
CATARACT - Glaucoma	0.002	0.991	0.372	0.376
CATARACT - Cataract	0.001	0.994	0.514	0.200
CATARACT - Normal	0.011	0.961	0.423	0.250

The results that they are shown in Table 23 confirm the robustness of EvoPruneDeepTL. This insight is the same as in the previous table: low values of Hamming distance in the best chromosomes of EvoPruneDeepTL implies high values of CKA in the closest element. Moreover, the Hamming distance from the elements of pruning consecutive layers is higher than in the other case, but the CKA values are also higher. This is a fair result because the difference in accuracy between EvoPruneDeepTL and the reference models is lower than in the other case. Note that the class Cataract from this dataset is the class with the fewest gap in accuracy, and this is shown in its CKA value. The Glaucoma class is the opposite of Cataract, and the CKA value is lower. In all the cases, EvoPruneDeepTL confirms the ability to learn the neurons that they are indispensable to achieve a greater performance, and that is the main difference between the reference models.

Table 23: Comparison of the CKA measure for pruning consecutive layers in CATARACT-Class.

	Pruning consecutive layers				
Dataset	# Layer	Hamming Distance EvoPruneDeepTL	$CKA_{Closest}$	Hamming Distance No Pruned Model	CKA_{Ref}
CATARACT - Retina	Layer 1 Layer 2	0.020	0.980 0.983	0.567	0.685 0.769
CATARACT - Glaucoma	Layer 1 Layer 2	0.001	0.996 0.997	0.556	0.680 0.745
CATARACT - Cataract	Layer 1 Layer 2	0.001	0.998 0.998	0.240	0.891 0.916
CATARACT - Normal	Layer 1 Layer 2	0.027	0.975 0.979	0.469	0.784 0.843

The second dataset under analysis is RPS. Table 24 shows the results of the feature selection models for RPS-Class. We see that the pair Hamming distance and CKA of the closest have a great result in two of the three cases. Moreover, the results of the other metrics achieve a great results, similarly to CATARACT-Class with this model. For that reason, we confirm that, for this model, EvoPruneDeepTL is also able to learn the neurons that maximize the accuracy for the problem.

A special case is RPS-Paper. When RPS does not have this class, the problem seems to be a very easy task, because all the models in the previous experiments for this dataset achieve the maximum accuracy. This is the only case in which the CKA for the best and its closest element is lower in comparison with the others. This is due to the fact that the problem at hand can be solved with many chromosomes, as they all have the maximum accuracy value, so the chromosomes might not be very similar, because the range of possible solutions is wide.

Table 24: Comparison of the CKA measure for feature selection in RPS-Class.

	Feature Selection			
Dataset	Hamming Distance EvoPruneDeepTL	$CKA_{Closest}$	Hamming Distance No Pruned Model	CKA_{Ref}
RPS - Paper	0.199	0.416	0.485	0.224
RPS - Rock	0.020	0.926	0.507	0.195
RPS - Scissors	0.040	0.863	0.766	0.104

Table 25 shows the pruning of consecutive layers that it is performed by EvoPruneDeepTL. Similar conclusions are obtained from these results. First, we see that the Paper class has the same problem which appears in the previous table. However, the other two groups of experiments are harder to solve and this have been drawn in the CKA of the closest element, because both layers have a great value of this measure. In the counterpart, the CKA values for the reference also has a similar understanding, which belongs to the fact that EvoPruneDeepTL pruning of consecutive layers learns the best neurons for both layers.

In overall, EvoPruneDeepTL is also a robust model for this dataset both in feature selection and pruning consecutive layers, and it also proves that the difference in accuracy between our models and the reference models is stated in the CKA_{Ref}.

Table 25: Comparison of the CKA measure for pruning consecutive layers in RPS-Class.

	Pruning consecutive layers				
Dataset	# Layer	Hamming Distance EvoPruneDeepTL	$CKA_{Closest}$	Hamming Distance No Pruned Model	CKA_{Ref}
RPS - Paper	Layer 1 Layer 2	0.217	0.725 0.761	0.486	0.764 0.825
RPS - Rock	Layer 1 Layer 2	0.032	0.977 0.983	0.492	0.757 0.852
RPS - Scissors	Layer 1 Layer 2	0.010	0.982 0.974	0.667	0.597 0.668

The last dataset in this section is PAINTING. The first table relates to the feature selection model of EvoPruneDeepTL. Table 26, once more, shows that the lowest Hamming distance of the best chromosome when it is compared with its closest, brings high values of CKA. The conclusion is clear, EvoPruneDeepTL is robust. Moreover, the values of CKA_{Ref} are also a good estimation of how EvoPruneDeepTL looks for the best neurons. Both CATARACT and PAINTING have lots of similarities in the feature selection model.

Table 26: Comparison of the CKA measure for feature selection in PAINTING-Class.

Dataset	Feature Selection			
	Hamming Distance EvoPruneDeepTL	$CKA_{Closest}$	Hamming Distance No Pruned Model	CKA_{Ref}
PAINTING - Sculpture	0.002	0.992	0.471	0.259
PAINTING - Painting	0.003	0.986	0.502	0.202
PAINTING - Iconography	0.002	0.993	0.433	0.236
PAINTING - Engraving	0.006	0.984	0.441	0.291
PAINTING - Drawings	0.021	0.925	0.432	0.236

Table 27 shows the results for pruning consecutive layers in the dataset PAINTING. These results, again, prove that the closest and best elements of EvoPruneDeepTL achieve a great value of CKA given a low value of Hamming distance, which is the best output that we can have. Moreover, the CKA values for the reference models are high, but this is due to the fact that the difference in accuracy between the models is lower. However, this also proves that EvoPruneDeepTL also learns the best neurons for this problem.

Table 27: Comparison of the CKA measure for pruning consecutive layers in PAINTING-Class.

Dataset	Pruning consecutive layers				
	# Layer	Hamming Distance EvoPruneDeepTL	$CKA_{Closest}$	Hamming Distance No Pruned Model	CKA_{Ref}
PAINTING - Sculpture	Layer 1 Layer 2	0.001	0.998 0.999	0.359	0.832 0.879
PAINTING - Painting	Layer 1 Layer 2	0.004	0.998 0.998	0.578	0.701 0.777
PAINTING - Iconography	Layer 1 Layer 2	0.001	0.998 0.998	0.701	0,593 0.672
PAINTING - Engraving	Layer 1 Layer 2	0.001	0.999 0.999	0.255	0.895 0.931
PAINTING - Drawings	Layer 1 Layer 2	0.003	0.998 0.998	0.427	0.797 0.874

In this section, we have shown that EvoPruneDeepTL is able to capture the relevance of the diverse classes and datasets that they are shown. Thanks to that adaption, EvoPruneDeepTL has shown its robustness and its ability to search for the best neurons to tackle the problem at hand.

5.6.2. Effects of a gradual aggregation of a class in the problem at hand

This section is devised to analyze the impact of a class when it appears as a new class in a dataset, and it increases its number of examples over time. We have selected the class Iconography of PAINTING for these

experiments. The reason lies in the fact that the class with a low percentage of examples is a minority class of this dataset, but it becomes the majority class of PAINTING when all the examples are used. Adding more examples of this class lets us check how EvoPruneDeepTL is able to adapt to different scenarios for the same dataset when a class is gradually growing on its importance in the dataset.

This section has a similar structure to the previous one. First, we show the results of the experiments for pruning consecutive layers and feature selection models of EvoPruneDeepTL. Then, the CKA values is also presented to perform the same double comparison as it has been done in the last section. For this section, the notation for the dataset is PAINTING-Iconography we talk about the dataset resulting from adding more examples, and in the tables this is shown as PAINTING-Pct%, where $Pct = 20, 40, 60$ and 80.

The first experiments we show are in Table 28. These experiments have been carried out with the dataset PAINTING with the different percentage, as we have previously explained. The results show that, as the percentage of data increases, the models tend to become better. If we compare these results with those obtained with the full data set, we see that the model with 80% of the data (and with 60% of the data also for feature selection) is the closest to the full model (see Table 6 and Table 12).

Reviewing the results tables with the full dataset and comparing them with these results, we observe that for the consecutive layer pruning model, the number of active neurons is lower in these experiments. The same phenomenon occurs in most of the feature selection cases, except when 40% of the data is used, where this number increases as the model improves the accuracy for that dataset at the cost of increasing the percentage of active neurons.

Table 28: Average results for PAINTING-Iconography with pruning consecutive layers and feature selection.

Dataset	Measure	EvoPruneDeepTL Both	No Pruning Both	EvoPruneDeepTL Feature Selection	No Pruning Feature Selection
PAINTING - 20%	Accuracy	0.944	0.931	0.945	0.826
	% Active neur.	47	100	56	100
PAINTING - 40%	Accuracy	0.945	0.929	0.950	0.826
	% Active neur.	39	100	65	100
PAINTING - 60%	Accuracy	0.946	0.931	0.954	0.839
	% Active neur.	53	100	51	100
PAINTING - 80%	Accuracy	0.947	0.927	0.953	0.839
	% Active neur.	41	100	47	100
PAINTING Full	Accuracy	0.953	0.939	0.958	0.939
	% Active neur.	51	100	55	100

Next, we follow the same process as for the previous section, in which the different classes of various data sets were analyzed. We show the value of CKA for the feature selection model and for the pruning consecutive layers model. Table 29 shows the results for the different feature selection models applied to the various data percentage options of the Iconography class. The CKA value, which compares the best with its closest element in the history of the execution, is very high. This implies that EvoPruneDeepTL is a robust model, since the phenotype obtained from the genotype is very similar. In addition, the other CKA value reported by the reference model indicates that EvoPruneDeepTL is capable of the neurons important for the model, thus explaining the difference in accuracy between the two approaches.

Table 29: Comparison of the CKA measure for feature selection in PAINTING-Iconography.

Dataset	Feature Selection			
	Hamming Distance EvoPruneDeepTL	$CKA_{Closest}$	Hamming Distance No Pruned Model	CKA_{Ref}
PAINTING - 20%	0.001	0.995	0.442	0.277
PAINTING - 40%	0.007	0.976	0.352	0.461
PAINTING - 60%	0.019	0.934	0.484	0.205
PAINTING - 80%	0.001	0.998	0.530	0.177

The following table, Table 30. The robustness of the proposal becomes evident when comparing the best element with its closest element in each of the runs, which has also occurred in the previous case. CKA values are extremely high when these elements are compared.

The comparison with respect to the reference models shows results similar to those of other cases of consecutive layer pruning. The difference in accuracy is reflected in the CKA, which is higher than in the feature selection cases, because this difference is larger when it comes to the pruning of the extracted features of the network.

Table 30: Comparison of the CKA measure for pruning consecutive layers in PAINTING-Iconography.

Dataset	Pruning consecutive layers				
	# Layer	Hamming Distance EvoPruneDeepTL	$CKA_{Closest}$	Hamming Distance No Pruned Model	CKA_{Ref}
PAINTING - 20%	Layer 1 Layer 2	0.001	0.999 0.998	0.529	0.732 0.794
PAINTING - 40%	Layer 1 Layer 2	0.001	0.998 0.999	0.607	0.677 0.755
PAINTING - 60%	Layer 1 Layer 2	0.001	0.999 0.999	0.466	0.758 0.828
PAINTING - 80%	Layer 1 Layer 2	0.001	0.998 0.998	0.594	0.694 0.762

This section has allowed us to see EvoPruneDeepTL in different situations it has had to face. From data sets with fewer classes so that our proposal is able to adapt to all the subclasses that compose it (first subsection), to the gradual increase of a class from being the minority to the majority (second subsection). The study which has been performed in this section relies on a measure which allows us to study the robustness of EvoPruneDeepTL and, in addition, allows us to see the differences in accuracy of the models that they have been developed.

The results in both sets of experiments show that the stochasticity that might be present in the proposal is not influential. The results of the CKA measure when comparing the best trained network found by EvoPruneDeepTL and its closest trained network for each of the runs show the high degree of robustness of EvoPruneDeepTL.

The comparison of EvoPruneDeepTL with networks with all neurons active (CKA_{Ref}) shows us a twofold conclusion. When we are dealing with the feature selection models, those that performed the pruning of the extracted features of the network, the difference in accuracy is reflected in the value of CKA, which is very low and that means that the models are very different On the other hand, for the case of pruning consecutive layers, the CKA_{Ref} value reflects models with less difference in comparison to the previous case. However, both in feature selection and pruning consecutive layers it is observed that EvoPruneDeepTL is able to search for the neurons that best approximate the problem to be solved.

6. Advantages and disadvantages of EvoPruneDeepTL

This section is devised to discuss the advantages and disadvantages of EvoPruneDeepTL, considering the diverse and large experimentation which has been done in the manuscript. The advantages of EvoPruneDeepTL can be summarized as follows:

- **Specialization of the last layers of networks.**

  An important element of EvoPruneDeepTL is the transfer learning. This is one of the most commonly used techniques. We have refined its process, which is the extraction of pre-trained features and then, the specialization of the fully-connected layers. In this context, EvoPruneDeepTL, and specifically the GA which is composed of, when applied to these layers does not limit the network learning compared to other evolutionary models in the literature that require high computational time to evaluate the datasets, as they train the whole network. For that reason, EvoPruneDeepTL can be applied to more complex datasets.

- **Performance over reference models and efficient pruning methods from the literature.**

  The usage of an evolutionary model that focuses on pruning neurons of the fully-connected layers achieves a better performance than other pruning methods when applied under the same conditions of EvoPruneDeepTL. The positive effect of the genetic algorithm is the selection of the best neurons of these layers, so that the evolution towards the best configuration for the networks is obtained thanks to EvoPruneDeepTL.

- **Constructive modeling over the last layers of the networks.**

  In the different experiments that have been carried out in the sections, we have observed that performing the pruning constructively based on the number of layers achieves good results. Pruning one-layer networks achieves good results, but when the number of layer increases, it is shown that performing the pruning over a single layer of two-layer networks improves the one-layer networks. Nonetheless, the simultaneous pruning of the both layers achieves a better modeling of the datasets than all the previous pruning models.

- **Pruning the extracted features of the network against pruning fully-connected layers.**

  EvoPruneDeepTL obtains better results by pruning the self-generated features resulting from transfer learning versus pruning the fully-connected layers. This is an intuitive idea because the learned patterns or features are different for each problem, and the learned features for the original problem may not be useful for the target problem. Knowing which characteristics matter is crucial to the problem at hand. The evolutionary process allows pruning these features and selecting those that best solve the modeling problem under consideration.

- **Generalization of EvoPruneDeepTL to other feature extractors.**

  One of the advantages of EvoPruneDeepTL is that the model is generalizable to diverse feature extractors. This is an important advantage because EvoPruneDeepTL is able to achieve a great performance over different datasets and with diverse networks. These are used to extract the features of the dataset, thanks to the transfer learning technique. EvoPruneDeepTL has improved the reference models and pruning models from the literature in both the pruning of two consecutive layers and the pruning of the features extracted by the networks.

- **Adaptation to relevant classes and gradual aggregation of data for the problem at hand.**

  EvoPruneDeepTL has achieved a great performance in different situations. However, it is also important to see how it adapts to different situations within the datasets themselves. EvoPruneDeepTL has improved the performance of the reference models, but a measure is needed to support the quality of these models. However, this increase in performance has been proven by CKA not to be the result of chance, so EvoPruneDeepTL models are able to search for the best neurons to maximize the accuracy of the problem.

- **The good results of EvoPruneDeepTL do not depend on randomness.**

  The evolutionary algorithm in which EvoPruneDeepTL lies in is a stochastic algorithm and the results may be affected by randomness. This is a risk about the model, so it is required to check if the good results are biased by the randomness. For that reason, we have introduced a new measure, CKA. This measure compares the differences between the pruned networks, and it is a way to measure the robustness of EvoPruneDeepTL. The values of the CKA for the various experiments of the previous section shed light on the fact that EvoPruneDeepTL results do not depend on the randomness.

The main disadvantage of EvoPruneDeepTL is:

- **Execution time of EvoPruneDeepTL.**

  The main drawback of EvoPruneDeepTL it is the time that is required to execute the model. In comparison with the pruning methods from the literature and the reference models, the table of execution times (see Table 11) shows the speed of the other models, but EvoPruneDeepTL is slower than these models. The time difference is made up by improved network performance, thanks to the usage of EvoPruneDeepTL. Nonetheless, there are some practical cases in which the training time is not a problem, like in medical diagnosis, because the main objective is obtaining a better percentage of the models in terms of accuracy.

7. Conclusions

This paper has introduced EvoPruneDeepTL, a novel model that sparsifies the architecture of the last layers of a DL model initialized using TL. EvoPruneDeepTL is a combination of sparse layers and EA, so that the neurons of these layers are pruned using the EA, in order to adapt them to the problem to tackle and deciding which neurons/connections to leave active or inactive.

EvoPruneDeepTL is a flexible model that evolves models with one and two layers and even two layers at the same time. Our results show that the pruning over complete neurons is better than pruning connections individually, establishing the last one as the best encoding strategy. The evolution of the sparse layer improves these models in terms of accuracy and also in terms of complexity of the network. In comparison with compared reference models and pruning methods from the literature, EvoPruneDeepTL achieves a better performance than all of them. The choice of one among the pruning models or feature selection has been answered and informed with experimental evidence: the FS scheme derived from EvoPruneDeepTL has shown a better performance, in most cases, than the pruning methods. The ability of adaptation of EvoPruneDeepTL to other feature extractors has been tested. Lastly, EvoPruneDeepTL has also shown its capability to adapt to the relevance of diverse problems and it has also achieved an outstanding level of robustness, which implies that the results do not depend on random nature of the search operators used by the GA that lies at the core of the proposed evolutionary pruning method.

From an overarching perspective, this work aligns with a growing strand of contributions where evolutionary computation and DL have synergized together to yield evolved models that attain better levels of performance and/or an increased computational efficiency. Indeed, this fusion of concepts (forged as Evolutionary Deep Learning) has been used for other evolution processes, including hyperparameter or structural tuning. Another recent case of the symbiosis of EA's and DL are represented in AutoML-Zero that use an evolutionary search to automatically search the best DL structure. AutoML-Zero and EvoPruneDeepTL

are two great examples of the benefits of combining EA's and DL that outline the potential and promising path of successes envisioned for this research area.

Future research work stemming from the results reported in this study is planned from a two-fold perspective. To begin with, we plan to achieve larger gains from the combination of DL and EA by extending the evolutionary search over higher layers of the neural hierarchy, increasing the number of evolved layers and neurons per layer. To this end, we envision that exploiting the layered arrangement in which neurons are deployed along the neural architecture will be essential to ensure an efficient search. The second research line relates to this last thought, aiming to improve the search algorithm itself by resorting to advanced concepts in evolutionary computation (e.g. niching methods or co-evolutionary algorithms).

Finally, a third research path arising from this work is the reformulation of the pruning problem and the adaptation of EvoPruneDeepTL to select which activation functions to be used in the fully connected part of the neural network. Our hypothesis is that the pruning problem can be reformulated so that every decision variable represents which activation function to utilize in every neuron. Efforts will be invested in this direction, adapting operators within EvoPruneDeepTL to efficiently explore the combinatorial search space of the reformulated optimization problem.

Acknowledgments

F. Herrera, D. Molina and J. Poyatos are supported by the Andalusian Excellence project P18-FR-4961, the infrastructure project with reference EQC2018-005084-P and the R&D and Innovation project with reference PID2020-119478GB-I00 granted by the Spain's Ministry of Science and Innovation and European Regional Development Fund (ERDF). Aritz D. Martinez and Javier Del Ser would like to thank the Basque Government for the funding support received through the EMAITEK and ELKARTEK programs, as well as the Consolidated Research Group MATHMODE (IT1456-22) granted by the Department of Education of this institution.

References

Aneja, N., & Aneja, S. (2019, July). Transfer learning using cnn for handwritten devanagari character recognition. In *1st International Conference on Advances in Information Technology (ICAIT), Chikmagalur, India, 2019*.

Anwar, S., Hwang, K., & Sung, W. (2017). Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, *13*, Article 32. https://doi.org/10.1145/3005348.

Assunção, F., Lourenço, N., Machado, P., & Ribeiro, B. (2019). Denser: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines*, *20*, 5–35. https://doi.org/10.1007/s10710-018-9339-y.

Back, T., Fogel, D. B., & Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*. (1st ed.). IOP Publishing Ltd.

Chambers, L. D. (2000). *The Practical Handbook of Genetic Algorithms: Applications*. (2nd ed.). Chapman and Hall/CRC.

[dataset]Sungjoon Choi (2020). Cataract dataset. Retrieved from https://www.kaggle.com/jr2ngb/cataractdataset. Accessed September 10, 2020.

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *The Journal of Machine Learning Research*, *20*, 1997–2017. https://dl.acm.org/doi/10.5555/3322706.3361996.

Elsken, T., Metzen, J. H., & Hutter, F. (2019, May). Efficient multi-objective neural architecture search via lamarckian evolution. In *7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 2019*.

Fernandes, C., & Rosa, A. (2001, May). A study on non-random mating and varying population size in genetic algorithms using a royal road function. In *Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, 2001*.

Frankle, J., & Carbin, M. (2019, May). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 2019*.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. (1st ed.). Addison-Wesley Longman Publishing Co., Inc.

[dataset] Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A., & Herrera, F. (2019). Coral species identification with texture or structure images using a two-level classifier based on convolutional neural networks. *Knowledge-Based Systems*, *184*, Article 104891. https://doi.org/10.1016/j.knosys.2019.104891.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. (1st ed.). MIT Press.

Gretton, A., Bousquet, O., Smola, A., & Schölkopf, B. (2005, October). Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory, Singapore, 2005*.

Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A., Krawczyk, B., & Herrera, F. (2019). Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation. *Expert Systems with Applications*, *118*, 315 – 328. https://doi.org/10.1016/j.eswa.2018.10.010.

Han, S., Mao, H., & Dally, W. (2016, May). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2016*.

Han, S., Pool, J., Tran, J., & Dally, W. (2015, Dececember). Learning both weights and connections for efficient neural network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal Canada, 2015*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016, June). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016*.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017, July). Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017*.

Iba, H. (2018). *Evolutionary Approach to Machine Learning and Deep Neural Networks: Neuro-Evolution and Gene Regulatory Networks*. (1st ed.). Springer.

Iguyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182. https://dl.acm.org/doi/10.5555/944919.944968.

Jung, S., Park, J., & Lee, S. (2019, May). Polyphonic sound event detection using convolutional bidirectional lstm and synthetic data-based transfer learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 2019*.

Khan, S., Islam, N., Jan, Z., Din, I. U., & Rodrigues, J. J. C. (2019). A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognition Letters*, *125*, 1–6. https://doi.org/10.1016/j.patrec.2019.03.022.

Kokiopoulou, E., Hauth, A., Sbaiz, L., Gesmundo, A., Bartók, G., & Berent, J. (2020, August). Task-aware performance prediction for efficient architecture search. In *24th European Conference on Artificial Intelligence, Santiago de Compostela, Spain, 2020*.

Kornblith, S., Norouzi, M., Lee, H., & Hinton, G. (2019, June). Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning, California, CA, USA, 2019*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012, December). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems, Lake Tahoe, NV, USA, 2012*.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324. https://doi.org/10.1109/5.726791.

Lee, H., Pham, P., Largman, Y., & Ng, A. Y. (2009, December). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems, Vancouver, B.C., Canada. 2009*.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., & Murphy, K. (2018, September). Progressive neural architecture search. In *European Conference on Computer Vision, Munich, Germany 2018*.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., & Kavukcuoglu, K. (2018, April). Hierarchical representations for efficient architecture search. In *6th International Conference on Learning Representations (ICLR), Vancouver, B.C., Canada. 2018*.

Liu, J., Wang, Y., & Qiao, Y. (2017, February). Sparse deep transfer learning for convolutional neural network. In *31st AAAI Conference on Artificial Intelligence (AAAI), San Francisco, CA , USA*.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11–26. https://doi.org/10.1016/j.neucom.2016.12.038.

Liu, Z., Sun, M., Zhou, T., Huang, G., & Darrell, T. (2019, May). Rethinking the value of network pruning. In *7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 2019*.

Long, X., Ben, Z., & Liu, Y. (2019a). A survey of related research on compression and acceleration of deep neural networks. *Journal of Physics: Conference Series*, *1213*, Article 052003. https://doi.org/10.1088/1742-6596/1213/5/052003.

Long, X., Ben, Z., Zeng, X., Liu, Y., Zhang, M., & Zhou, D. (2019b). Learning sparse convolutional neural network via quantization with low rank regularization. *IEEE Access*, *7*, 51866–51876. https://doi.org/10.1109/ACCESS.2019.2911536.

Lu, Z., Sreekumar, G., Goodman, E., Banzhaf, W., Deb, K., & Boddeti, V. (2021). Neural architecture transfer. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *43*, 2971–2989. https://doi.org/10.1109/TPAMI.2021.3052758.

Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., & Banzhaf, W. (2019, July). NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 2019*.

Luo, J.-H., Wu, J., & Lin, W. (2017, October). ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. In *IEEE International Conference on Computer Vision, Venice, Italy, 2017*.

Mantzaris, D., Anastassopoulos, G., & Adamopoulos, A. (2011). Genetic algorithm pruning of probabilistic neural networks in medical disease estimation. *Neural Networks*, *24*, 831 – 835. https://doi.org/10.1016/j.neunet.2011.06.003.

Martín, A., Lara-Cabrera, R., Fuentes-Hurtado, F., Naranjo, V., & Camacho, D. (2018). Evodeep: a new evolutionary approach for automatic deep neural networks parametrisation. *Journal of Parallel and Distributed Computing*, *117*, 180–191. https://doi.org/10.1016/j.jpdc.2017.09.006.

Martinez, A. D., Del Ser, J., Villar-Rodriguez, E., Osaba, E., Poyatos, J., Tabik, S., Molina, D., & Herrera, F. (2021). Lights and shadows in evolutionary deep learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges. *Information Fusion*, *67*, 161–194. https://doi.org/10.1016/j.inffus.2020.10.014.

Masson, H., Bhuiyan, A., Nguyen-Meidine, L. T., Javan, M., Siva, P., Ayed, I. B., & Granger, E. (2021). Exploiting prunability for person re-identification. *EURASIP Journal on Image and Video Processing*, *2021*, Article 22. https://doi.org/10.1186/s13640-021-00562-6.

Mehdipour Ghazi, M., Yanikoglu, B., & Aptoula, E. (2017). Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing*, *235*, 228 – 235. https://doi.org/10.1016/j.neucom.2017.01.018.

Mohapatra, R., Saha, S., Coello, C. A. C., Bhattacharya, A., Dhavala, S. S., & Saha, S. (2022). Adaswarm: Augmenting gradient-based optimizers in deep learning with swarm intelligence. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *6*, 329–340. https://doi.org/10.1109/TETCI.2021.3083428.

[dataset]Laurence Moroney (2019). Rock, paper, scissors dataset. Retrieved from http://www.laurencemoroney.com/rock-paper-scissors-dataset/. Accessed September 10, 2020.

Muhammad, K., Khan, S., Ser, J. D., & Albuquerque, V. H. C. d. (2021). Deep learning for multigrade brain tumor classification in smart healthcare systems: A prospective survey. *IEEE Transactions on Neural Networks and Learning Systems*, *32*, 507–522. https://doi.org/10.1109/TNNLS.2020.2995800.

Muhammad, K., Ullah, A., Lloret, J., Ser, J. D., & de Albuquerque, V. H. C. (2020). Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, *22*, 4316–4336. https://doi.org/10.1109/TITS.2020.3032227.

Nibali, A., He, Z., & Wollersheim, D. (2017). Pulmonary nodule classification with deep residual networks. *International journal of computer assisted radiology and surgery*, *12*, 1799–1808. https://doi.org/10.1007/s11548-017-1605-6.

Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H. G., & Ogata, T. (2015). Audio-visual speech recognition using deep learning. *Applied Intelligence*, *42*, 722–737. https://doi.org/10.1007/s10489-014-0629-7.

Pan, S., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*, 1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019, January). Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 2019*.

Real, E., Liang, C., So, D., & Le, Q. (2020, July). Automl-zero: evolving machine learning algorithms from scratch. In *International Conference on Machine Learning, 2020*.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., & Kurakin, A. (2017, August). Large-scale evolution of image classifiers. In *International Conference on Machine Learning, Sydney, Australia, 2017*.

Roy, D., Murty, K. S. R., & Mohan, C. K. (2015, July). Feature selection using deep neural networks. In *2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 2015*.

[dataset]Virtual Russian Museum (2018). Art images: Drawing/painting/sculptures/engravings. Retrieved from https://www.kaggle.com/thedownhill/art-images-drawings-painting-sculpture-engraving, Accessed September 10, 2020.

Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., & McCool, C. (2016). Deepfruits: A fruit detection system using deep neural networks. *Sensors*, *16*, Article 1222. https://doi.org/10.3390/s16081222.

Salehinejad, H., & Valaee, S. (2021). Edropout: Energy-based dropout and pruning of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, (pp. 1–14). In Press https://doi.org/10.1109/TNNLS.2021.3069970.

Samala, R., Chan, H.-P., Hadjiiski, L., Helvie, M., Richter, C., & Cha, K. (2018). Evolutionary pruning of transfer learned deep convolutional neural network for breast cancer diagnosis in digital breast tomosynthesis. *Physics in Medicine and Biology*, *63*, Article 095005. https://doi.org/10.1088/1361-6560/aabb5b.

Scott, G. J., England, M. R., Starms, W. A., Marcum, R. A., & Davis, C. H. (2017). Training deep convolutional neural networks for land–cover classification of high-resolution imagery. *IEEE Geoscience and Remote Sensing Letters*, *14*, 549–553. https://doi.org/10.1109/LGRS.2017.2657778.

Shin, H.-C., Roth, H., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., & Summers, R. (2016). Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, *35*, 1285–1298. https://doi.org/10.1109/TMI.2016.2528162.

Simonyan, K., & Zisserman, A. (2015, May). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, San Diego, CA, USA, 2015*.

[dataset] Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S., & Batra, N. (2020, May). Plantdoc: A dataset for visual plant disease detection. In *7th ACM IKDD CoDS and 25th COMAD, Hyderabad, India, 2020*.

Srinivas, S., & Babu, R. V. (2015, September). Data-free parameter pruning for deep neural networks. In *Proceedings of the British Machine Vision Conference (BMVC), Swansea, UK, 2015*.

Stanley, K., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*, 99–127. https://doi.org/10.1162/106365602320169811.

Sultana, F., Sufian, A., & Dutta, P. (2018, November). Advancements in image classification using convolutional neural network. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, India, 2018*.

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., & Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, *35*, 1299–1312. https://doi.org/10.1109/TMI.2016.2535302.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018, October). A survey on deep transfer learning. In *International Conference On Artificial Neural Networks (ICANN), Rhodes, Greece, 2018*.

[dataset]Hafiz Tayyab Rauf, Saleem, B. A., Lali, M. I. U., Khan, M. A., Sharif, M., & Bukhari, S. A. C. (2019). A citrus fruits and leaves dataset for detection and classification of citrus diseases through machine learning. *Data in Brief*, *26*, Article 104340. https://doi.org/10.1016/j.dib.2019.104340.

Ullrich, K., Welling, M., & Meeds, E. (2017, April). Soft weight-sharing for neural network compression. In *5th International Conference on Learning Representations (ICLR), Toulon, France, 2017*.

Wan, S., Liang, Y., & Zhang, Y. (2018). Deep convolutional neural networks for diabetic retinopathy detection by image classification. *Computers and Electrical Engineering*, *72*, 274–282. https://doi.org/10.1016/j.compeleceng.2018.07.042.

Wang, Z., Li, F., Shi, G., Xie, X., & Wang, F. (2020). Network pruning using sparse learning and genetic algorithm. *Neurocomputing*, *404*, 247 – 256. https://doi.org/10.1016/j.neucom.2020.03.082.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*, Article 9. https://doi.org/10.1186/s40537-016-0043-6.

Wen, L., Li, X., Li, X., & Gao, L. (2019, May). A new transfer learning based on vgg-19 network for fault diagnosis. In *IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD), Porto, Portugal, 2019*.

Yildirim, O., Pławiak, P., Tan, R.-S., & Acharya, U. (2018). Arrhythmia detection using deep convolutional neural network with long duration ecg signals. *Computers in Biology and Medicine*, *102*, 411–420. https://doi.org/10.1016/j.compbiomed.2018.09.009.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014, December). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems, Montreal, Canada, 2014*.

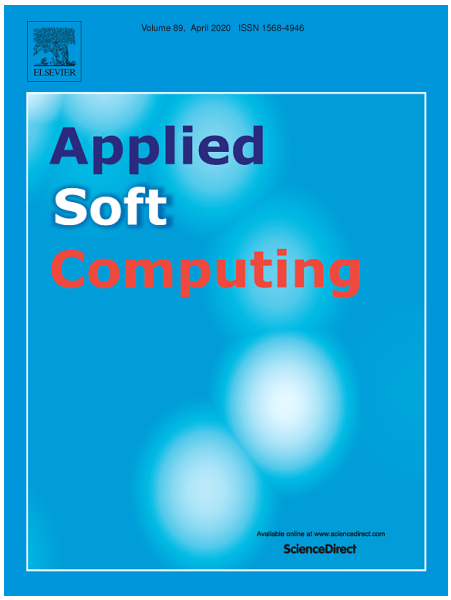
Zhou, X., Gong, W., Fu, W., & Du, F. (2017, May). Application of deep learning in object detection. In *IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), Wuhan, China, 2017*.

Zhu, M., & Gupta, S. (2018, April). To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations (ICLR), Vancouver, B.C. , Canada, 2018*.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. (2018, June). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018*.

Zou, Q., Ni, L., Zhang, T., & Wang, Q. (2015). Deep learning based feature selection for remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters*, *12*, 2321–2325. https://doi.org/10.1109/LGRS.2015.2475299.

3 Multiobjective evolutionary pruning of Deep Neural Networks with Transfer Learning for improving their performance and robustness



- Journal: Applied Soft Computing
- JCR Impact Factor: 8.7
- Rank: 21/145
- Quartile: Q1
- Category: Computer Science, Artificial Intelligence
- Status: Published

Ref.: Poyatos, J., Molina, D., Martínez-Seras, A., Del Ser, J., & Herrera, F. (2023). Multiobjective evolutionary pruning of Deep Neural Networks with Transfer Learning for improving their performance and robustness. Applied Soft Computing, 147, 110757. DOI: https://doi.org/10.1016/j.asoc.2023.110757.

Multiobjective Evolutionary Pruning of Deep Neural Networks with Transfer Learning for improving their Performance and Robustness

Javier Poyatos[a], Daniel Molina[a,*], Aitor Martínez-Seras[b], Javier Del Ser[b,c], Francisco Herrera[a]

[a]*Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, Granada, 18071, Spain*
[b]*TECNALIA, Basque Research & Technology Alliance (BRTA), Derio, 48160, Spain*
[c]*University of the Basque Country (UPV/EHU), Bilbao, 48013, Spain*

Abstract

Evolutionary Computation algorithms have been used to solve optimization problems in relation with architectural, hyper-parameter or training configuration, forging the field known today as Neural Architecture Search. These algorithms have been combined with other techniques such as the pruning of Neural Networks, which reduces the complexity of the network, and the Transfer Learning, which lets the import of knowledge from another problem related to the one at hand. The usage of several criteria to evaluate the quality of the evolutionary proposals is also a common case, in which the performance and complexity of the network are the most used criteria. This work proposes MO-EvoPruneDeepTL, a multi-objective evolutionary pruning algorithm. MO-EvoPruneDeepTL uses Transfer Learning to adapt the last layers of Deep Neural Networks, by replacing them with sparse layers evolved by a genetic algorithm, which guides the evolution based in the performance, complexity and robustness of the network, being the robustness a great quality indicator for the evolved models. We carry out different experiments with several datasets to assess the benefits of our proposal. Results show that our proposal achieves promising results in all the objectives, and direct relation are presented among them. The experiments also show that the most influential neurons help us explain which parts of the input images are the most relevant for the prediction of the pruned neural network. Lastly, by virtue of the diversity within the Pareto front of pruning patterns produced by the proposal, it is shown that an ensemble of differently pruned models improves the overall performance and robustness of the trained networks.

Keywords- Evolutionary Deep Learning, Multi-Objective Algorithms, Pruning, Out of Distribution Detection, Transfer Learning

*Corresponding author
Email addresses: jpoyatosamador@ugr.es (Javier Poyatos), dmolina@decsai.ugr.es (Daniel Molina), aitor.martinez@tecnalia.com (Aitor Martínez-Seras), javier.delser@tecnalia.com (Javier Del Ser), herrera@decsai.ugr.es (Francisco Herrera)

1. Introduction

Evolutionary Computation (EC) refers to a family of global optimization algorithms inspired by biological evolution (Back & Schwefel, 1996, May). EC algorithms such as Evolutionary Algorithms (EA) (Back et al., 1997) have been used to solve several complex optimization problems which cannot be analytically solved in polynomial time. In many real-world optimization problems, there is not only one criterion or objective to improve, but several objectives to consider. Multi-Objective Evolutionary Algorithms (MOEAs) are an family of EAs capable of efficiently tackle optimization problems comprising several goals (Deb, 2011).

Structural search and, in particular, Neural Architecture Search (NAS), is one of the non-polynomial problems which has been approached with EAs over the years (Martinez et al., 2021). This problem consists of looking for neural network configurations that fit better one dataset by optimizing the performance or loss of the network in function of the selected evaluation metric (Stanley & Miikkulainen, 2002). There have been several Neural Networks (NN), and particularly when integrated with Deep Learning (DL) called as Deep Neural Network (DNN), to which NAS has been applied are well-known networks with one or more objectives (Pham et al., 2018, July; Yang et al., 2020, June).

Among other decision variables considered in NAS, this area has also approached the improvement of a NN by optimally pruning their neural connections. Pruning techniques seek to reduce the number of parameters of the network, targeting network architectures with less complexity. Usually this comes at the cost of a lower performance of the network. When a new learning task is present, a manner to compensate the lack of quality of data is the usage of the Transfer Learning (TL), whose most straightforward approximation is the usage of pre-trained network in very large datasets (Krizhevsky et al., 2017) for the extraction of features, followed by a specialization of the last layers of the network. Due to the fact that the number of trainable parameters of these last layers is lower, it is possible to avoid an early overfitting of the network, which can happen if there are few examples for large models. For those cases, the search of optimal pruning patters using evolutionary NAS is done for these last layers (Poyatos et al., 2023).

Robustness is one of the unavoidable requirements to ensure proper performance in scenarios where risks must be controlled and certain guarantees are needed to ensure the proper performance of the models (ISO, 2021a,b). The combination of EAs together with techniques that allow evaluating the robustness of the models paves the way towards the creation of better models for all types of problems. It could be useful to incorporate robustness as a target, but unfortunately, robustness has been rarely considered an objective (Wang et al., 2021). Robustness can be measured in several ways for a DNN model, one being the performance in Out-of-Distribution (OoD) detection problems (Hendrycks & Gimpel, 2016). This problem consists of detecting whether a new test instance queried to the model belongs to the distribution underneath the learning dataset i.e., the In-Distribution (InD) dataset or, instead, it belongs to another different distribution (correspondingly, the Out-Distribution dataset, OoD).

The natural extension of NAS is the development of proposals with several objectives. In this scenario, the MOEAs can take place, as they evolve the networks meanwhile an optimization of several objectives is made (Lu et al., 2022b; Elsken et al., 2019). The MONAS term arises as the union of MO algorithms which are used for NAS problems (MONAS). MONAS algorithms usually rely in several objectives, being a standard objective the performance of the network. The complexity of the network is a common second objective, which can be modeled as the number of parameters pruned form the network, network compression or other alternatives. More sophisticated proposals consider another objective based on the energy consumption or hardware device in use, among others (Chitty-Venkata & Somani, 2022; Wei et al., 2022). The addition of the robustness, with a OoD detection technique applied to the DL model being optimized, as an additional objective unleashes a new vision for the MONAS proposals.

The main hypothesis is the convenience of using a MOEA to evolve the pruning patterns of the fully-connected layers of a neural network via a sparse representation, simultaneously according to the generalization performance of the network, its complexity and the robustness of a OoD detection technique relying on the activation signals inside the network against samples that may or may not belong to the distribution of the training data.

This work finds its inspiration in the recent work in (Poyatos et al., 2023), in which dense layers are pruned using a configuration that define the active neurons. In the previous work, that configuration is

evolved by using a binary genetic algorithm guided by the performance of the network. In this manuscript, the previous problem is reformulated to optimize the pruning patterns with a MOEA, in which the search is guided by the three previously mentioned objectives. Intuitively, a highly-pruned network may reduce its performance and the robustness of an OoD detection method that relies on the activations of the pruned network. For that reason, a minimum fraction of neurons must be active (i.e. non-pruned) to achieve balanced models with good balance (in the Pareto sense) between performance and robustness.

In this context, OoD detection falls within the umbrella of the Open-World Learning (OWL) paradigm (Parmar et al., 2022; Zhou, 2022). OWL pursues models that are capable of learning in non-controlled environments, so that models become increasingly knowledgeable as they are queried with new data. However, OWL can also be considered one of the technologies supporting General Purpose Artificial Intelligence (GPAI), which is largely enabled by AI generating AI models (Clune, 2019; Real et al., 2020, July). Since this work proposes a MOEA to optimize DL models, it can be regarded as an example of AI enhancing AI.

In detail, this work proposes an approach based on the evolution of the pruning patterns of fully-connected layers using a MOEA, which we hereafter refer to as Multi-Objective Evolutionary Pruning for Deep Transfer Learning (MO-EvoPruneDeepTL). The goal of MO-EvoPruneDeepTL is to search for the best pruning patterns in the last layers of the NN to adapt them to the problem at hand. To accomplish this task, MO-EvoPruneDeepTL utilizes several techniques. To begin with, TL allows for the extraction of features by leveraging pretrained neural models, so that the specialization of the target NN takes place in the last fully-connected layers of the network hierarchy. At this point of the network pruning is as suitable mechanism to prune non-important features that do not contribute to the flow of information throughout the last part of the NN, which connects pretrained features to the output to be predicted. MOEA then emerges as an efficient method to solve the problem of finding good pruning patterns according to the aforementioned different objectives: performance, complexity and, robustness of the network. To measure the robustness of a model, an OoD detection technique is used, which is based on the capability of the model to detect unseen data in the training step. Ideally, robust models with good performance and low complexity should be desirable. However, the fact that pruning affects the activations throughout the last stage of the network causes that performance and robustness can be affected by the pruning intensity imposed by any given pruning pattern. This conflicting nature of the objectives under consideration is the rationale for seeking the optimal set of pruning patterns that best balance between them by using a MOEA. Finally, we will show that a byproduct of the estimated Pareto front is that NNs pruned by patterns belonging to the front can be combined together, yielding an ensemble model with increased performance and/or robustness with respect to any of its compounding NNs. This exposes that the pruning solutions give rise to NN models that present a sufficient diversity to improve their performance in accuracy and robustness over different value ranges of the objectives driving the search.

To assess the quality of MO-EvoPruneDeepTL, different experiments have been designed that allow inspecting several aspects of the performance of MO-EvoPruneDeepTL from different perspectives. To that end, the main purpose of the experimental setup is to provide an informed answer to the following research questions (RQ):

(**RQ1**) How are the approximated Pareto fronts produced by the proposal in each of the considered datasets?

(**RQ2**) Is there any remarkable pruning pattern that appears in all the solutions of the Pareto front?

(**RQ3**) Do our models achieve an overall improvement in performance when combined through ensemble modeling?

A general insight about these experiments is the the achievement of optimized networks in these objectives, but also that the evolutionary process gives rise to pruning patterns that maintain relevant neurons with information about the input of the model, and leads to the use of ensembles to further improve modeling performance in terms of generalization and robustness to OoD.

The rest of the article is structured as follows: Section 2 briefly overviews background literature related to the proposal. Section 3 shows the details of the proposed MO-EvoPruneDeepTL model. Section 4 presents the experimental framework designed to thoroughly examine the behavior of MO-EvoPruneDeepTL with

respect to the RQ formulated above. In Section 5, we show and discuss in depth the results obtained by MO-EvoPruneDeepTL in the different experiments. Several indicators are presented to show the quality of MO-EvoPruneDeepTL. Finally, Section 6 draws the main conclusions from this study, as well as future research lines stimulated by our findings.

2. Related work

The aim of this section is to make a review of contributions to the literature about the key elements of this study: Neural Architecture Search (Subsection 2.1), Transfer Learning and Pruning of Convolutional Neural Networks (CNN, Subsection 2.2) and OoD detection (Subsection 2.3). The last paragraph of this section resumes the benefits of MO-EvoPruneDeepTL.

2.1. Neural architecture search

The design of the NN that best fits for the problem at hand is a challenging task. The search for the best design of the network is also considered as another problem, as it is necessary to find the best architecture that optimally fits the data. In this context, NAS has achieved a great importance in this area. The main purpose of NAS proposals is the search for the best design of the NN to solve the considered problem.

First NAS-based proposals started to emerge in the beginning of this century. NEAT, presented in (Stanley & Miikkulainen, 2002) was a pioneering proposal about how EAs — specifically, a Genetic Algorithm (GA) – can be used to evolve NNs. They showed that a constructive modeling of the NN with the benefits of the GA can lead to optimized NN topologies. The natural extension of this seminal work allowing for the evolution of DNN was presented years after in (Miikkulainen et al., 2019), in which the authors use a co-evolutionary algorithm based on the co-operation scheme to evolve DNN.

In the last years, more NAS proposals have been developed. One of them is EvoDeep (Martín et al., 2018), in which the authors create an EA with specific operators to create and evolve DL models from scratch. More examples of the importance of NAS come with the next proposals. In (Dufourq & Bassett, 2017, November), authors propose another EA to perform the evolution of NN, similarly to the previous proposal, but with a difference in relation to the fitness function, which is influenced by the accuracy and complexity of the network. The other example is presented in (Assunção et al., 2019). In this case, the evolution comes in two different ways: topology and parameters of the Convolutional Neural Networks (CNN). In (Trivedi et al., 2018), the authors propose a GA that evolves the weights of the softmax layer to improve the performance of the NN. Suganuma et al. propose in (Suganuma et al., 2020) a $(1 + \lambda)$ evolutionary strategy to evolve DNN. In 2020, (Real et al., 2020, July) presents an advanced technique that automatically searches for the best model, operating from scratch and obtaining a good performance with the problems at hand. The use of NAS has been applied in other areas like the Reinforcement Learning (RL). In that area, there is a great example of NAS (Zoph & Le, 2016). In that work, authors use a recurrent network (RNN) to generate the model descriptions of NN and train this RNN with RL to maximize the expected accuracy of the generated architectures on a validation set.

There are more examples of NAS in the literature like the NAS algorithm which comprises of two surrogates through a supernet, with the objective of improving the gradient descent training efficiency (Lu et al., 2020, August). Another NAS comes in (Lu et al., 2022a), in which the authors propose a pipeline with also a surrogate NAS applied to real-time semantic segmentation. They manage to convert the original NAS task into an ordinary MO optimization problem.

Lastly, there are more advanced techniques of NAS and EA given by (Real et al., 2019, January), in which a new model for evolving a classifier is presented, and by (Real et al., 2020, July), in which the authors propose AutoML-Zero, an evolutionary search to build a model from scratch (with low-level primitives for feature combination and neuron training) which is able to get a great performance over the addressed problem.

The main characteristic of the previous NAS proposals is the evolution of the DL model guided by a single objective, usually the accuracy or another that measures the performance of the network. The following proposals share a common aspect: the evolution of the model is done using more than one objective. This leads to the algorithms in the field of MONAS.

We can find several approaches of MONAS that have been applied to diverse fields with great results. One of them is presented in (Elsken et al., 2019, May). This work proposes a MONAS that lets the approximation of the Pareto-front of all the architectures. In relation to medical images area, in (Baldeon-Calisto & Lai-Yuen, 2020) a MONAS that evolves both accuracy and model size is proposed. Moreover, following this research in medical images, in (Baldeon Calisto & Lai-Yuen, 2020), the authors use a MO evolutionary based algorithm that minimizes both the expected segmentation error and number of parameters in the network. Another interesting work is presented in (Calisto & Lai-Yuen, 2020), in which they have created a pipeline for the automatic design of neural architectures while optimizing the network's accuracy and size.

Typically, MONAS evaluate two or three objectives. A common objective is usually the performance of the network. The others objectives are related with the complexity of the network and other empirical and measurable objectives. In (Lu et al., 2021), the authors propose a MOEA for the design of DNN for image classification, adopting the classification performance and the number of floating-point operations as its objectives. Another example is DeepMaker, (Loni et al., 2020), which is a MOEA approach that considers both the accuracy of the network and its size to evolve robust DNN architectures for embedded devices.

There are some well-known MOEAs in the literature. One of them is NSGA-II. A new version of it has been developed to use it for NAS (Lu et al., 2019, July), called NSGA-Net. This proposal looks for the best architecture through a three-step search based on an initialization step, followed by an exploration step that performs the EA operators to create new architectures, and an exploitation step that uses the previous knowledge of all the evaluated architectures.

2.2. Transfer learning and pruning

One of the objectives that EAs used for NAS usually aim to optimize is the complexity of the network. NN are structures with a great amount of parameters. These networks are composed of two main parts. The first one extracts the main features of the problem, i.e., learns to distinguish the patterns of the images (when working with image classification) and the second part is responsible to classify these patterns into several classes.

In this context, TL appears as a figure that helps in the learning process when there are few data, i.e., prevents the overfitting when the input examples is not large (Pan & Yang, 2010). TL is a DL mechanism encompassing a broad family of techniques. The most common method of TL with DL is the usage of a previous network structure with pre-trained parameters in a similar problem to the related task, being trained with huge datasets like (Krizhevsky et al., 2017). This fact involves the usage of a DL model with fixed and pre-trained weights in the convolutional layers with a dataset and then add and train several layers to adapt the network to a different classification problem (Khan et al., 2019).

Another technique to reduce the complexity of the networks is pruning. Pruning a CNN model consists of reducing the parameters of the model, but it may lead into a decrease of the performance of the model. Several approaches to prune networks have been developed over the years, such as (Han et al., 2015, December; Srinivas & Babu, 2015, September). These methods have been already used in several problems, rendering great performance.

An example of the fusion of EAs, DNN and pruning is shown in (Wang et al., 2020), which proposes a novel approach based on a combination of pruning CNN of sparse layers (layer with fewer connections between neurons) guided by a GA. The main consequence of this study is the reduction of a great fraction of the network, but at the penalty of a lower generalization performance of the network.

Following the idea of EAs and DNN, in (Poyatos et al., 2023) the authors propose also propose a combination of sparse layers and a GA. They have shown that pruning can be done in a TL scheme with sparse layers and EAs. Their proposal is only guided by the performance of the model, but they also achieve a great reduction in the optimized sparse layers.

2.3. Out of distribution detection

Robustness is a term that has been used with related yet different meanings among the literature of the Machine Learning (ML) community. In this work, we refer to the model's ability to handle the unknown, to detect whether it has been queried with an example of a not learned distribution, therefore refusing to make the classification it has been trained to do. This is precisely what the OoD detection framework measures.

In this problem, a model learns to classify instances in the different classes from a training dataset that is sampled from a distribution, namely the InD. After the process, the model is asked to correctly distinguish between test examples that are drawn equally from either the InD or from a semantically different distribution, the OoD dataset (Yang et al., 2021). The term semantically different refers to the fact that the classes contained in this foreign distribution are distinct from the ones present in the InD. As ML and DNN model are not natively prepared for this task, an OoD detection technique is wrapped around the model to allow this behavior. Typically, these techniques are based on creating a score for every example processed by the model, such that the score obtained by an OoD instance is significantly different from the one obtained by a InD example. Then, by simply defining a threshold on this score, the model can decide whether an instance is from the in or out distribution.

A great variety of methods exist in the literature, which was started by Hendrycks & Gimpel (2016), where the so-called *baseline* method was introduced. It relies on the simple observation that InD instances tend to have greater Maximum Softmax Probability, the softmax probability of the predicted class. By simply applying a threshold to this score, they achieved acceptable performance on many classification problems. In Liang et al. (2018, Aprila), this idea was refined by applying temperature scaling to the softmax probabilities, what further separates apart from each other the distributions of the scores of the in- and out- distribution samples probabilities. Authors also implemented an input preprocessing pipeline that enhanced a bit the performance by adding a small quantity of gradient and softmax dependent noise. The paper presented in Lee et al. (2018), instead of using the softmax probabilities, exploits the feature space of the layer right before softmax and assumes that it follows a multivariate Gaussian distribution, enabling the calculation of its mean and variance for every sample. After creating a class-conditional distribution utilizing the training samples, the score for every test sample is the closest Mahalanobis distance between the sample and the calculated Gaussian class-conditional distributions.

The technique proposed in (Hendrycks et al., 2019, May), in contrast to previous works, focuses on modifying model's training by adding a term to the loss function (that depends on the classification of the task, density estimation, etc.), helping the model learn heuristics that will improve the performance of other OoD methods applied afterwards. This new term needs to be trained with OoD data, which can be obtained by leveraging the large amount of data publicly available on the internet. Authors prove that the learned heuristics for arbitrary OoD datasets generalize well to other unseen OoD data. Thereafter, (Liu et al., 2020) based its detector in what they called the free energy function, that combines concepts of the energy-based models with modern neural networks and their capability of assigning a scalar to every instance fed to the model without changing its parametrization. Specifically, the free energy function is based on the logits of the network, and the work empirically demonstrates that OoD instances tend to have higher energy, enabling the distinction between InD and OoD data. In the following work in (Lin et al., 2021, June) exploited the idea that easy OoD samples can be detected by leveraging low-level statistics. On this basis, several intermediate classifiers are trained at different depths and each example is outputted through one of them depending on its complexity. To measure complexity, a function based on the number of bits used to encode the compressed image is harnessed. The OoD scoring function employed is the above presented energy function adapted to the corresponding depth.

Although only a few research contributions are presented in this work, it must be noted that the OoD problem has been widely studied in the literature (Salehi et al., 2021), with proposals ranging from the more complex and well performing ones to the more simple yet effective ones. As the aim of this paper is to show that the robustness in the OoD can be affected when the pruning of the network is done. Therefore, the OoD detection method will be selected to be computationally cheap yet effective, to not add computational complexity to the MOEA.

In this section, a review of the related work from three different perspectives has been presented. Terms like MONAS, MOEA are important as this work presents a new work about these topics. Moreover, it is based on a TL scheme in which an evolutionary pruning of the last layers is done. In the last years, several proposals have been published over these topics, i.e, MOEAs that search for the best architecture attending to one or more objectives and also pruning approaches for CNNs. However, this study introduces a new manner to guide the evolutionary pruning of the models with the usage of a OoD mechanism. MO-EvoPruneDeepTL tries to solve the problem of achieving robust models with high performance and least

active neurons. This scheme, a MOEA that performs pruning in the last layers (TL paradigm) with three objectives is a new contribution to all this fields at the same time.

3. Multi-Objective Evolutionary Pruning for Deep Transfer Learning

This section is devised to explain the details of MO-EvoPruneDeepTL. First, in Subsection 3.1 the formulation to the problem at hand is presented. In Subsection 3.2, we will describe the objectives used to guide the search. Then, in Subsection 3.3, the description of the OoD detector is explained. The DL and network schemes are shown in Subsection 3.4. Finally, in Subsection 3.5, the evolutionary parts of MO-EvoPruneDeepTL are described.

3.1. Problem formulation

This section aims at defining and explaining the mathematical components that circumscribe MO-EvoPruneDeepTL. We explore different concepts needed to fully understand the basics of our study.

We define the concept of dataset. Mathematically, we define a training dataset $\mathcal{D} \doteq \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ composed by N (instance, label) pairs. Such a dataset is split in training, validation and test partitions, such that $\mathcal{D} = \mathcal{D}_{tr} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$ with $|\mathcal{D}_{tr}| = N_{tr}$.

Another important concept to keep in mind is the model. We define a model M_θ to represent the relationship between its input $\mathcal{X}$ and its corresponding output $y \in \{1, \ldots, Y\}$, where Y denotes the number of classes present in $\mathcal{D}$. Learning the parameter values θ^* that best model this relationship can be accomplished by using a learning algorithm $\theta^* = \text{ALG}(M; \mathcal{D}_{tr})$ that aims to minimize a measure of the difference (*loss*) between the model's output and its ground truth over the training instances in $\mathcal{D}_{tr}$ (e.g. gradient back-propagation in neural networks). In what follows M_θ is assumed to be a NN, so that θ represent the totality of trainable weights in its neural connections.

In the context of TL for classification tasks, the NN M_θ is assumed to be composed by a pre-trained feature extractor $F_\phi(\mathbf{x})$ (whose parameters ϕ are kept fixed while ALG$(\cdot)$ operates), and a dense (i.e. fully-connected) part $G_\theta(\cdot)$ that maps the output of the feature extractor to the class/label to be predicted. Therefore, after tuning the trainable parameters of the network as $\theta^* = \text{ALG}(G; \mathcal{D}_{tr})$, the class $\widehat{y} \in \{1, \ldots, Y\}$ predicted for an input instance $\mathbf{x}$ is given by:

$$\widehat{y} = (F \circ G_{\theta^*})(\mathbf{x}) = G_{\theta^*}(F(\mathbf{x})), \tag{1}$$

where $\circ$ denotes composition of functions. When predictions are issued over the validation partition $\mathcal{D}_{val}$, a measure of accuracy can be done by simply accounting for the ratio of correct predictions to the overall size of the set, i.e. $\text{ACC}_{val} = (1/N_{val}) \cdot \sum_{i \in \mathcal{D}_{val}} \mathbb{I}(\widehat{y}_i = y_i)$, where $\mathbb{I}(\cdot)$ equals 1 if its argument is true (0 otherwise).

Bearing this notation in mind, pruning can be defined as a binary vector $\mathbf{p} = \{p_j\}_{j=1}^{P}$, where P denotes the length of the feature vectors extracted by $F_\phi(\mathbf{x})$ for every input instance to the network. As such, $p_j = 0$ indicates that neural links that connect the j-th component of the feature vector to the rest of neurons in the dense part $G_\theta(\cdot)$ of the network are disconnected, causing that all trainable parameters from this disconnected input to the output of the overall model are pruned. Conversely, if $p_j = 1$ the j-th input neuron is connected to the densely connected layers of the neural hierarchy. By extending the previous notation, the training algorithm is redefined to $\theta^*(\mathbf{p}) = \text{ALG}(G; \mathcal{D}_{tr}, \mathbf{p})$ to account from the fact that the network has been pruned as per $\mathbf{p}$. This dependence of the trained parameters on the pruned vector propagate to the measure of accuracy over the validation instances, yielding $\text{ACC}_{val}(\mathbf{p})$. Likewise, a measure of the reduction of the number of trainable parameters can be also computed for a given pruning vector $\mathbf{p}$ relative to the case when no pruning is performed (i.e., $\mathbf{p} = \mathbf{1} \doteq \{1\}_{j=1}^{P}$) as $\text{MEM}(\mathbf{p}) = |\theta(\mathbf{p})|/|\theta(\mathbf{1})|$.

Intuitively, a good pruning strategy should consider the balance between the reduced number of trainable parameters and its impact on the accuracy when addressing the modeling task at hand. Reducing the amount of parameters to be stored has practical benefits in terms of memory space, and can yield a lower inference latency when the trained model is queried.

A third dimension of the network that can be affected by pruning is its capacity to detect OoD instances. A significant fraction of the techniques proposed so far for identifying query samples that deviate from the distribution of training data rely on the network dynamics between neurons while the instance flows through the network. This is the case of ODIN (Liang et al., 2018, Aprila), BASELINE (Hendrycks & Gimpel, 2016) and ENERGY (Liu et al., 2020), among others. To quantify the capability of a pruned network $M_{\theta^*(\mathbf{p})}$ to detect OoD instances, we utilize other datasets $\boldsymbol{\mathcal{D}}' = \{\mathcal{D}'_d\}_{d=1}^{D_{OoD}}$ different from $\mathcal{D}$, whose instances $(\mathbf{x}', y')$ are assumed to be representative of the OoD test instances with which the model can be queried in practice. An OoD detection technique $T_{OoD}(\mathbf{x}; M_{\theta^*(\mathbf{p})}) \equiv T_{OoD}(\mathbf{x})$ processes the activations triggered by $\mathbf{x}$ throughout the trained pruned model $M_{\theta^*(\mathbf{p})}$ so as to decide whether $\mathbf{x}$ is an InD ($T_{OoD}(\mathbf{x}) = 0$) or an OoD instance (corr. $T_{OoD}(\mathbf{x}) = 1$). This being said, true positive and false negative rates can be computed for $T(\mathbf{x})$ over the test subset $\mathcal{D}_{test}$ of $\mathcal{D}$ and random N_{val}/D_{OoD}-sized samples drawn from every other dataset $\mathcal{D}'_d$, which can be aggregated in a compound performance metric. Among other choices for this purpose, we consider the AUROC measure AUROC($\mathbf{p}$), which measures the ability of $T(\cdot)$ to discriminate between positive and negative examples. This measure is set dependent on $\mathbf{p}$ in accordance with previous notation, as $T(\mathbf{x})$ operates on the neural activations stimulated by $\mathbf{x}$.

3.2. Objectives of MO-EvoPruneDeepTL

This section introduces the objectives that guide MO-EvoPruneDeepTL during its evolutionary process. We define them using the notation previously commented in Subsection 3.1.

The optimization problem addressed in this work aims to discover the set of Pareto-optimal pruning vectors $\{\mathbf{p}_k^{opt}\}_{k=1}^{K}$ that best balance between three objectives:

1. The modeling performance of the pruned model over dataset $\mathcal{D}$. This performance is measured with the accuracy over the test dataset ($\mathcal{D}_{test}$). It is the percentage of well classified images out of the total set of images.

2. The number of active neurons left after the pruning operation. The number of active neurons corresponds with the remaining active connections after the pruning and evolutionary process.

3. The capability of an OoD detection technique to discriminate between OoD and InD data by inspecting the activations inside the pruned model.

Mathematically:

$$\{\mathbf{p}_k^{opt}\}_{k=1}^{K} = \arg_{\mathbf{p}\in\{0,1\}^P} \left[\max \mathrm{ACC}_{val}(\mathbf{p}), \min \mathrm{MEM}(\mathbf{p}), \max \mathrm{AUROC}(\mathbf{p})\right], \tag{2}$$

$$\text{s.t.} \quad \mathcal{D} : \text{In-distribution dataset}, \tag{3}$$

$$\mathcal{D}'_1, \ldots, \mathcal{D}'_{D_{OoD}} : \text{Out-of-distribution datasets}, \tag{4}$$

$$F_\phi(\mathbf{x}) : \text{Pre-trained feature extractor}, \tag{5}$$

$$T(\mathbf{x}) : \text{Out-of-distribution detection technique}. \tag{6}$$

3.3. Out of Distribution detector of MO-EvoPruneDeepTL

In the following subsection the technique selected to assess the OoD performance of the pruned models is presented, along with a clarification about the metrics used to measure it.

Due to the fact that every new child of the population in the evolutionary algorithm must have its OoD performance correctly assessed, the chose method should not entail a big computational burden while maintaining a sufficient effectiveness in detecting OoD samples. The technique presented in (Liang et al., 2018, Aprilb), ODIN, fulfills these requirements and is the selected one.

Before explaining ODIN, the already mentioned performance metric used in this study must be clarified, namely the AUROC or *Area Under the Receiver Operation Characteristic curve*. It is a threshold-independent metric for binary classification that can be considered as the probability that the model ranks a random positive example with higher score than a random negative example. Is defined as TPR/FPR,

which stand for True Positive Rate and False Positive Rate respectively and can be computed as TPR = TP/(TP + FN) and FPR = FP/(FP + TN). Therefore, in order to compute the AUROC, the FPR value for every TPR needs to be calculated. In this work, TP is used to refer to an in-distribution sample correctly classified as such, whereas a TN represents an OoD sample detected correctly by the OoD detector.

The basic principle of ODIN is to use maximum softmax probability with temperature scaling as the OoD score for every sample, defined by the expression

$$f_{ODIN}(\mathbf{x};T) = \max_i(S_i(\mathbf{x};T)) = S_{\hat{y}}(\mathbf{x};T), \tag{7}$$

where $S_i(x;T)$ is the softmax probability of the i^th class for the input instance $\mathbf{x}$, scaled by a temperature parameter $T \in \mathbb{R}^+$. This scaled softmax can be calculated as:

$$S_i(x;T) = \frac{\exp(h_i(\mathbf{x})/T)}{\sum_{j=1}^{N}\exp(h_i(\mathbf{x})/T)}. \tag{8}$$

where $h_i(\mathbf{x}) \mid i \in \{1,\ldots,Y\}$ are the logits, the values prior to the softmax activation function. Then, and in accordance with notation presented in Subsection 3.1, the OoD detection technique T_{OoD}, ODIN in this case T_{ODIN}, will output a 1 if the instance's score is below a defined threshold, indicating that is considered an out-of-distribution sample, outputting a 0 otherwise:

$$\mathbf{x} \text{ belongs to } \begin{cases} \text{in-distribution} & \text{if } T_{ODIN}(\mathbf{x};T;\lambda) = 0 \iff f_{ODIN}(\mathbf{x};T) \geq \lambda, \\ \text{out-distribution} & \text{if } T_{ODIN}(\mathbf{x};T;\lambda) = 1 \iff f_{ODIN}(\mathbf{x};T) < \lambda. \end{cases} \tag{9}$$

It is important to remark that ODIN also uses an input preprocessing pipeline to further improve its performance in the OoD detection problem, but that in this study it will be discarded for the sake of reducing the computational burden of the algorithm.

So, in order to implement ODIN, the below presented steps must be followed. First, the model M_θ must be trained using the training set $\mathcal{D}_{tr}$ of the in-distribution dataset $\mathcal{D}$. Then the logits of the instances of the test set $\mathcal{D}_{test}$ must be extracted for the sake of calculating the temperature scaled softmax outputs using the equation (8). The OoD score of each input instance $f_{ODIN}(\mathbf{x};T)$ will be the maximum of these scaled softmax outputs, i.e., the value corresponding to the predicted class, as expression (7) indicates.

Next, same operation must be repeated with the the out-of-distribution detection set, composed by samples drawn from every other dataset $\mathcal{D}'_d$ as indicated in Subsection 3.1. In this manner, we have created two distributions of OoD scores: one for the in-distribution samples of $\mathcal{D}_{test}$ and other for the out-of-distribution ones. Now, the threshold on the score for each TPR is defined by using the score distribution of test instances and equation (9). The corresponding FPR for each TPR is computed by employing the OoD distribution and the defined threshold, obtaining a set of [TPR, FPR] values that compose the ROC curve. Finally, from this curve the AUROC can be computed, therefore obtaining the desired robustness score for the model M_θ and in-distribution dataset $\mathcal{D}$.

In this study, in order to evaluate the robustness of each model, a practical approach is used, which involves the usage of the OoD detector with the other datasets that are not covered in training phase. However, the design of the algorithm accommodates any other dataset as an OoD evaluation dataset.

3.4. Network characteristics of MO-EvoPruneDeepTL

In this subsection, we introduce the characteristics of the used network in MO-EvoPruneDeepTL. In our study, we use the TL paradigm, i.e., the weights of the convolutional phase are imported and fixed from another trained network in a similar task. For that reason, the DL model we use works as a feature extractor. The images pass through the network and it extracts their main features. These features correspond with the neurons which are evolved by the evolutionary components of MO-EvoPruneDeepTL.

The chosen network for this study is ResNet-50. The output of this network is a vector of 2048 features or characteristics. They are used as the input for the last layers of the neural network. In our case, following the research in (Poyatos et al., 2023), the last part of the neural network is composed by an input layer that

receives an input vector of 2048 features (i.e., the output of the ResNet-50), followed by a hidden layer of 512 neurons and, finally, an output layer with as many neurons as classes defined in the problem at hand. This architecture is depicted in Figure 1, wherein *Layer 1* corresponds to our intermediate layer of 512 neurons, and *Network Features* denote the vector of 2048 features extracted from ResNet-50. We highlight in red the connections affected when a neuron is pruned. More specifically, each feature contributes to the neurons of the intermediate layer. The solver learns to distinguish which features are the most important and which are not, so that the whole set of connections from irrelevant features onward is eliminated, thereby not contributing to the rest of the intermediate layer.

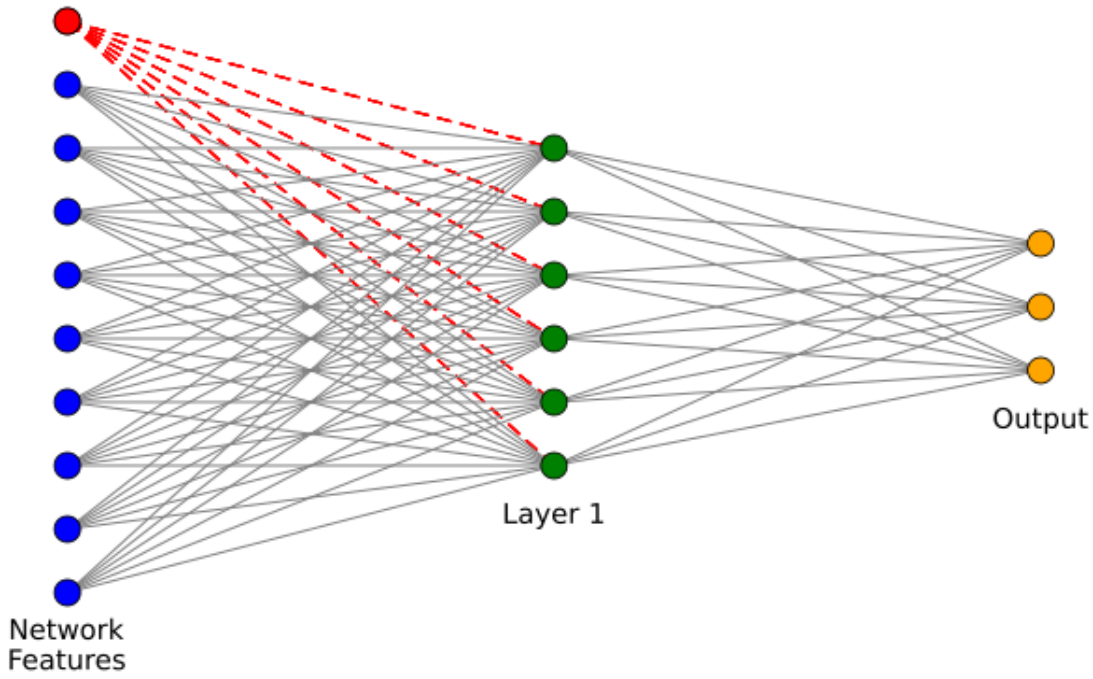


Figure 1: Pruning method of MO-EvoPruneDeepTL.

Following the previous idea, this network has fewer connections than a standard fully-connected layer, in which each neuron is connected to the next group of neurons. This type of layer is referred to as *sparse layer*. The genetic algorithm is in charge of finding an optimal pruning pattern for that sparse layer, in which the chromosome representing the pruning pattern can be decoded to an adjacency matrix. Figure 1 shows that this matrix defines the structural composition (connections) of the layer in the neural network. In particular, binary entries in the adjacency matrix correspond to the connections between the blue and green circles, so that certain connections will be removed (red lines) as a result of the pruning operation, rendering the sparse nature of the matrix and the layer itself.

3.5. Evolutionary components of MO-EvoPruneDeepTL

In this section, we introduce the evolutionary components of MO-EvoPruneDeepTL. It is a MOEA, called Non-Dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002). The population of networks is evolved using the common operators from this GA, but, in this case, only two individuals are used for the evolution as parents. As a result, two offspring individuals are produced. Our MO-EvoPruneDeepTL uses a binary encoding strategy, which represents if a neuron is active or not. A neuron is active if its gen is 1 or not active if it is 0. Thanks to this direct encoding approach, each gen determines uniquely a neuron in the decoded network.

The initialization of the chromosomes correspond with a random discrete initialization in $[0, 1]$, the selection is done using a binary tournament selection method, whereas the replacement strategy is the dual strategy of ranges of Pareto dominance and crowding distance of NSGA-II. Finally, the crossover and mutation operator are outlined:

Crossover: the crossover operator used in MO-EvoPruneDeepTL is the uniform crossover. This operator defines two new individuals from two parents. Mathematically, given these two parents **p** and **q**, where $\mathbf{p} = \{p_i\}_{i=1}^{P}$ and $\mathbf{q} = \{q_i\}_{i=1}^{P}$ and their length is P, the resultant offsprings **p'** $= \{p'_i\}_{i=1}^{P}$ and **q'** $= \{q'_i\}_{i=1}^{P}$ (also with length P) are generated using these equations:

$$\begin{aligned} p'_i &= \begin{cases} p_i & \text{if } r \leq 0.5 \\ q_i & \text{otherwise} \end{cases} \\ q'_i &= \begin{cases} q_i & \text{if } r \leq 0.5 \\ p_i & \text{otherwise} \end{cases} \end{aligned} \tag{10}$$

where r is the realization of a continuous random variable with support over the range $[0.0, 1.0]$. This operator creates two individuals using information of the genes of both parents. Each position i of the new individual takes the value of the gene of **p** or **q** until the offspring is fully created.

Mutation: the mutation performed by MO-EvoPruneDeepTL is the Bit Flip mutation. This operator needs a mutation probability defined by mut_p. Thus, for each chromosome, all of its genes can be mutated if the mutation is really performed, which means changing the value of the gene from active to not active or vice versa. The parameter that controls if a gene is flipped or not is mut_p.

Next, we give a brief explanation about the process that MO-EvoPruneDeepTL performs. First, we need to know the data required by MO-EvoPruneDeepTL, which is the dataset for training the network and its test dataset, the InD data, and also the OoD data, so that each model can also be tested on it. Lastly, the configuration of the GA and of the network are also required.

Once all the data is gathered, then the evolutionary process takes place. Algorithm 1 shows the pseudocode of MO-EvoPruneDeepTL. The beginning of the process is the standard procedure of initialization and evaluation of the initial population (lines 1 and 2). Then, the evolution is performed. In each generation, the operators are being executed sequentially. The parents are selected using the selection operator (line 4). After that, they generate their offspring using the crossover operator (line 5) which are mutated using the mutation operator (line 6). Then, both children are evaluated to obtain the values of the objectives that guide the evolutionary process. Thus, for each child, its chromosome is decoded into a sparse network (line 8) which is trained using the train set of the InD data (line 9). Then, the information contained in the logits is passed through the OoD detector which determines the robustness of the child using the AUROC metric (line 10). The accuracy is calculated using the test set of the InD (line 11) and the complexity of the network is also achieved using the number of neurons which are active in the child chromosome (line 12). Then, the objectives are retained as part of the information of the child for further generations (line 13).

4. Experimental framework

This section is intended to describe the framework surrounding the experiments conducted in this study. In Subsection 4.1, a detailed description of the datasets is given. Then, Subsection 4.2 shows the values of the parameters and the network setup of MO-EvoPruneDeepTL in the experiments.

4.1. Dataset information

In this study we have selected several datasets which fit in our working environment. These datasets represent a good choice for TL approaches due to their size, as the training and inference times are lower. Thus, these datasets are suitable for problems related with population metaheuristics, since a large number of individuals will be evaluated. We present a brief description of each dataset:

- CATARACT ([dataset]Sungjoon Choi, 2020) is a dataset related with the medical environment. It classifies different types of eye diseases.

Algorithm 1: MO-EvoPruneDeepTL

Input : InD dataset, OoD dataset, configuration of the GA and configuration of the network
Output: Evolved pruned network

```
1  Initialization of individuals of the population using the initialization operator;
2  Evaluation of the initial population (see lines 9-14);
3  while evaluations < max_evals do
4      Parent selection using binary tournament;
5      Generate offsprings using uniform crossover;
6      Mutation of individuals using the bit flip mutation;
7      for each child p in children population do
8          SparseNetwork_p ← Decodification of child chromosome;
9          SparseTrainedNetwork_p ← Train SparseNetwork_p using the train set of InD data;
10         AurocChild_p ← Robustness metric of child using OoD data;
11         AccChild_p ← Accuracy of SparseTrainedNetwork_p evaluated in test set of InD data;
12         ComplexChild_p ← Number of active neurons in SparseNetwork_p;
13         SolutionVector(AccChild,ComplexChild,AurocChild)_p;
14         evaluations+=1;
15     end
16     Replacement Strategy;
17 end
```

- LEAVES ([dataset]Hafiz Tayyab Rauf et al., 2019) is a dataset that is composed of images of different types of leaves, since healthy to unhealthy with different shades of green.
- PAINTING is related to the painting environment ([dataset]Virtual Russian Museum, 2018). This dataset is composed of images which represent different types of paintings.
- PLANTS is dataset which presents a great variety of leaves and plants, which ranges from tomato, or corn plants to other leaves, among others ([dataset] Singh et al., 2020, May).
- RPS ([dataset]Laurence Moroney, 2019) is a dataset whose purpose is to distinguish the gesture of the hands in the popular Rock Paper Scissors game from artificially-created images with different positions and skin colors.
- SRSMAS is based on the marine world whose aim is to classify different coral reef types ([dataset] Gómez-Ríos et al., 2019).

Next, we show some examples for several of the above datasets are shown in Fig. 2.

Finally, we highlight the main characteristics in quantitative terms of instances, classes and metrics with non-pruned networks for each dataset. Table 1 show these numbers.

4.2. Training and network setup

In this subsection, we describe both the training and network setup of MO-EvoPruneDeepTL. First, we explain how our datasets are split. Then, the network setup is presented. Lastly, we discuss the parameters of MO-EvoPruneDeepTL.

In this study, we use six different datasets in our experiments. We need to split the images of these datasets into a train and test subsets, as the evaluation of MO-EvoPruneDeepTL requires it. We have created a 5-fold cross-validation evaluation environment, meanwhile for the rest of the datasets, their train and test subsets had already been predefined.

Another component of MO-EvoPruneDeepTL is the used network along all the experiments. In our case, we have chosen ResNet-50 as the pre-trained network. We have selected ResNet-50 as the baseline feature

Figure 2: Images of datasets. Left: LEAVES examples. Middle: RPS examples. Right: SRSMAS examples.

Table 1: Datasets used in the experiments.

Dataset	**Image Size**	L (**# classes**)	**# Instances (train / test)**	**Accuracy (No Pruning)**	**AUROC (No Pruning)**
CATARACT	$(256, 256)$	4	480 / 121	0.732	0.870
LEAVES	$(256, 256)$	4	476 / 120	0.935	0.960
PAINTING	$(256, 256)$	5	7721 / 856	0.951	0.990
PLANTS	$(100, 100)$	27	2340 / 236	0.480	0.820
RPS	$(300, 300)$	3	2520 / 372	0.954	0.934
SRSMAS	$(299, 299)$	14	333 / 76	0.885	0.999

extraction method following up the conclusions drawn in (Poyatos et al., 2023), in which several experiments with other feature extractors such as DenseNet and VGG were found to perform worse than ResNet-50. Although other larger feature extractors may provide better performance, the choice of ResNet-50 is also related to the number of features obtained from the network, which directly influences the evaluation time of a solution. Based on these criteria, ResNet-50 is established as the pretrained backbone for our experiments, given its good balance between performance and complexity. This election has been taken to maintain the balance between the number of features, which leads to a higher computational space, and the performance obtained in the TL process. The combinatorial problem can be huge for typical values of feature extractors commonly used in problems where TL is in use. Using this network yields feature vectors $F_\phi(\mathbf{x})$ comprising $P = 2,048$ components, leading to a total of $2^{2,048} \approx 3.23 \cdot 10^{616}$ possible pruning patterns. Furthermore, the evaluation of pruned networks during the search requires repeatedly training over the instances in the test subset can be computationally expensive. Note that, although in our experiments a CNN model is used, the pruning can also be performed with other type or architectures, like Long Short-Term Memory (LSTM) (Wang et al., 2019).

These extracted features are passed through the last layers, which are the layers that are going to be trained. The model with the larger accuracy on the training set is saved. The optimizer of the training environment is the standard SGD. The parameters of MO-EvoPruneDeepTL are shown in Table 2. The

maximum number of training epochs is 600, but the training phase stops if no improvement is achieved in ten consecutive rounds. The last important parameter appears in the OoD phase. It is called $Temp_{ODIN}$ and it controls how the softmax values are computed using the logits from the Ind and OoD. The parameters of this study (see Table 2) have been selected by following recommendations of the authors. The values of the parameters controlling the genetic search operators have been taken from (Poyatos et al., 2023). The characteristics of the neural network are also those utilized in this previous work. Moreover, the OoD detection mechanism is based on (Liang et al., 2018, Aprila). In that work different temperature values were tested, reporting the value of the parameters of the technique that rendered the best results in their experiments. For this reason, in this work we have chosen the same value (namely, temperature equal to 1000).

Table 2: Parameters of MO-EvoPruneDeepTL.

Parameter	**Value**
Maximum Evals	200
# Runs	10
Population size	30
p_{mut}	$\frac{1}{P}$
Batch Size	32
Temp_{ODIN}	1000

The last contribution of this section is the discussion of the parameters of MO-EvoPruneDeepTL. The maximum evaluations of MO-EvoPruneDeepTL is set to 200 and the size of the population of networks for each generation is 30. Table 3 shows the evaluation time for each individual, so that the total time of execution is the time of the first column multiplied by the number of evaluations. Each OoD detection requires a minute, but in the datasets with the 5-fold cross-validation, this time reaches the five minutes. Moreover, we also indicate the inference time for test and the required time to calculate the AUROC metric in the OoD phase. Those times force us to keep a low number of runs and evaluations to meet a computationally affordable balance between the performance of our models and the high execution times required for our simulations. Moreover, although statistical tests are important to assess the significance of the differences in the results, but due to these limited number of runs, we can not apply them, as large number of runs is required to achieve statistically reliable insights.

Table 3: Average time in evaluations of MO-EvoPruneDeepTL.

Dataset	Total	Evaluation	Training and Inference	OoD Detection
CATARACT	332 min	1.66 min	0.66 min	1 min
LEAVES	1600 min	8 min	3 min	5 min
PAINTING	1700 min	8.5 min	7.5min	1 min
PLANTS	800 min	4 min	3 min	1 min
RPS	900 min	4.5 min	3.5 min	1 min
SRSMAS	1500 min	7.5 min	2.5 min	5 min

The experiments have been carried out using Python 3.6 and a Keras/Tensorflow implementation deployed and running on a Tesla V100-SXM2 GPU.

5. Results and discussion

This section is devised to analyze the behavior of MO-EvoPruneDeepTL. To this end, we define three research questions (RQ) which are going to be answered in the following subsections with diverse exper-

iments over the previous datasets. We will show and analyze several plots to illustrate the benefits of MO-EvoPruneDeepTL. The RQ can be stated as follows:

(**RQ1**) How are the approximated Pareto fronts produced by the proposal in each of the considered datasets?

The Pareto front can be defined as the set of non-dominated solutions, being chosen as optimal, if no objective can be improved without sacrificing at least one other objective. The problem at hand is approximated using a multi-objective approach. For that reason, we want to check that not only we have promising solutions in the extreme values of the Pareto front, but also to have a wide population of diverse solutions in the whole Pareto. As a consequence of that, to answer this RQ, we will analyze how is the Pareto front for each dataset and if there exists any direct connection between the objectives of the study: accuracy, complexity of the network, and robustness. In addition, a comparison to other pruning methods from the literature will be performed to check whether our proposal performs competitively against such methods.

(**RQ2**) Is there any remarkable pruning pattern that appears in all the solutions of the Pareto front?

We compare the pruning patterns of all the models of the Pareto fronts of MO-EvoPruneDeepTL to show if there are some important patterns which are key to identify the most important zones of the input images. We employ a well-known technique called Grad-CAM (Selvaraju et al., 2017, October), which uses the gradient of the classification score with respect to the convolutional features of the network to check which parts of the image are most important for the classification task. Grad-CAM lies in the group of Explainable Artificial Intelligence (XAI) techniques, as it produces details to make easy to understand which neurons are the relevant ones in all the experiments (Barredo Arrieta et al., 2020). These neurons lead to specific pixels or group of them of the original images that are passed through the network.

(**RQ3**) Do our models achieve an overall improvement in performance when merged through ensemble modeling?

MO-EvoPruneDeepTL trains a great variety of models which leads to a wide diversity of models in the Pareto front for each dataset. The aim of this RQ is to check whether the diversity of pruning patterns in the Pareto front can be used to improve our DL models through ensemble strategies. Our aim is to check if an ensemble of differently pruned models can yield more accurate predictions, leading to a better overall performance than their compounding models in isolation. Beyond improving the accuracy through ensembles, we will also explore whether ensemble modeling allows obtaining more robust models, so that the number of OoD samples that the network wrongly predicts as InD is lower.

This section is divided in Section 5.1, where we analyze the different Pareto front for each considered dataset in order to answer RQ1. Next, in Section 5.2, we will examine the different pruning patterns of our models. Precisely, we will look for the neurons that appears in most of them, and we will highlight the essential zones of the input images, as this is the key part to answer RQ2. Lastly, we will discuss in Section 5.3 the benefits of the diversity of our models when ensemble modeling is performed, to show if an improvement in terms of accuracy and AUROC is achieved, which the principles lines of the RQ3.

5.1. Answering RQ1: Analyzing the Pareto fronts of MO-EvoPruneDeepTL

The objective of this section is to answer RQ1 by performing a complete analysis of the Pareto fronts of MO-EvoPruneDeepTL and then, performing a comparison against competitive pruning methods of the literature. This analysis is to be performed focusing on two important aspects: i) how are the Pareto fronts for each dataset? and ii) how are the projections in each objective for each dataset? MO-EvoPruneDeepTL is run for 10 times, each yielding an estimation of the Pareto front between the three objectives. Such Pareto

front estimations contain solutions that dominate – in the Pareto sense – the rest of evaluated solutions during the evolutionary search. The elitist nature of the algorithm ensures that non-dominated solutions are retained in the population. Moreover, after the 10 executions of the algorithm, all Pareto front estimations are merged together. Non-dominated solutions in this merger compose a new Pareto front estimation (i.e., a *super* Pareto front) containing the best solutions found across the 10 runs of the algorithm. For simplicity, these solutions will be hereafter denoted as the Pareto front discovered by MO-EvoPruneDeepTL. Moreover, for each Pareto front, it has been included the results for the non-pruned network for each dataset, which are composed of solutions with all the active neurons and the accuracy and AUROC showed in Table 1.

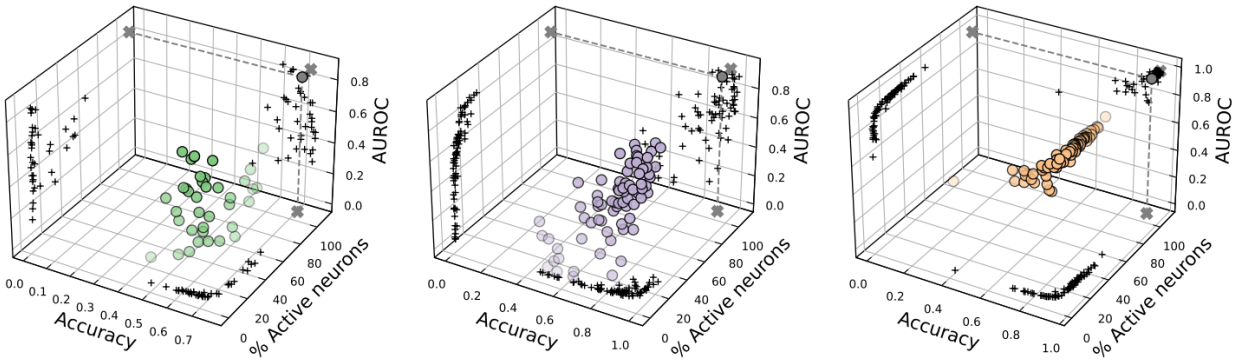


Figure 3: Pareto fronts of MO-EvoPruneDeepTL. Left: CATARACT dataset. Middle: RPS dataset. Right: PAINTING dataset.

With these graphics we analyze the quality of each Pareto and, particularly, by assessing the full spectrum of solutions that can be achieved in each of the Pareto. Moreover, we are going to study whether there is a direct relationship between any of the objectives we have formulated in the previous sections. In order to develop these plots, we have collected all the solutions of the super Pareto front (called Pareto front from now), selecting 10% of the best solutions for each objective in order to make their projections.

These Pareto front are presented in Figures 3 and 4. We can observe the diversity of pruning patterns produced by MO-EvoPruneDeepTL. Moreover, another insight from these Pareto front comes up when we inspect extreme values of each objective, as they systematically achieve good results in each dataset. Most of the solutions obtain high values of accuracy and robustness meanwhile their remaining active neurons are kept low.

First, we focus on the central part of the 3D projections, in which we visualize the three objectives. Our goal is to detect if there exist some kind of relationship between them. We clearly see that the projection in all the Pareto front takes values to the upper corner in which the three objectives present low values of percentage of active neurons, but high accuracy and AUROC. Moreover, this distribution of the points in both group of figures indicate that there is a tendency of the solutions to that plane in which the number of active neurons is low.

Analyzing the two-dimensional planes, there is not a clear relation between the performance and robustness. Nonetheless, the common point of this projection, namely, the complexity of the network, sheds light to the fact that it can be related with the performance and robustness separately. In both cases, a minimum number of active neurons is needed in order to start achieving good results in each objective. For both objectives, there is a certain range of optimal number of active neurons in which each of them obtains their best values.

The last experiment of this section addresses the performance comparison between MO-EvoPruneDeepTL and other competitive pruning methods from the related literature. In this work, we compare MO-EvoPruneDeepTL to the following two methods considered to be competitive counterparts for benchmarks between pruning proposals (Hoefler et al., 2021):

- `weight` (Han et al., 2015, December): The parameters with lower values are pruned at once. This method

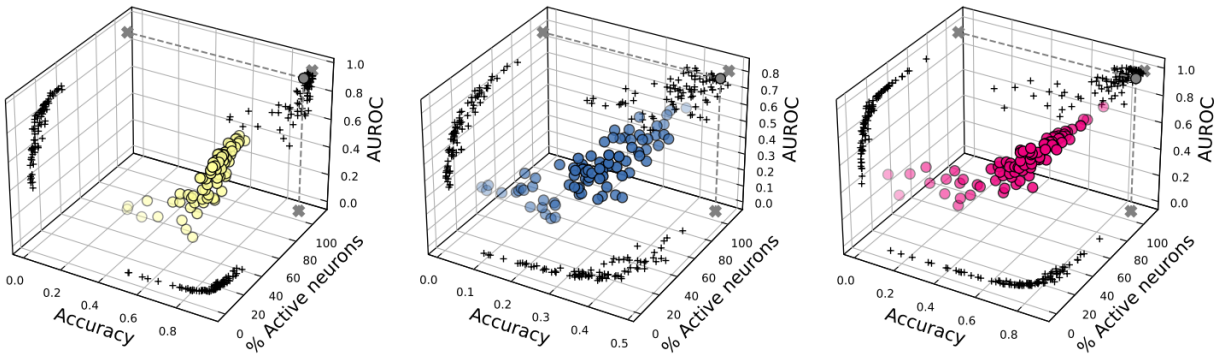


Figure 4: Pareto fronts of MO-EvoPruneDeepTL. Left: LEAVES dataset. Middle: PLANTS dataset. Right: SRSMAS dataset.

operates over the whole parameter set in the layer to be optimized.

- `neuron` (Srinivas & Babu, 2015, September): The neurons with lower mean input connection values are pruned.

Both pruning methods require a parameter that controls their execution, which is the target percentage of remaining neurons. This percentage represents the active weights remaining in the network that each of these methods reaches at the end of its execution. For a fair comparison, we force these methods to target the same percentage of remaining weights as in the solutions of the Pareto front estimated by MO-EvoPruneDeepTL. We note that the Pareto front estimation contains the non-dominated solutions found in the 10 runs of the algorithm. In this case, we have ordered the solutions in terms of complexity (second objective), which yields a distribution of ordered solutions, from least to most active weights. Based on this sorted list of solutions, we have chosen those with the median and the lowest values of the percentage of the remaining active weights (complexity of the network) to assess how MO-EvoPruneDeepTL behaves in these representative cases. Once we annotate these percentages, the pruning methods prune the fully-connected network until reaching such annotated values, giving rise to the accuracy values shown in the columns of this table.

Table 4 shows the results of the comparison between MO-EvoPruneDeepTL and the pruning methods. The second column indicates the target percentage of remaining weights corresponding to each dataset. The third, fourth, and fifth columns report the accuracy of weight pruning, neuron pruning, and MO-EvoPruneDeepTL, respectively. In addition, each dataset spans two rows in the table: the first row shows the median percentage of active weights and the accuracy of each of the proposals for that case, whereas the second row represents the case with the lowest percentage of active weights and their respective levels of accuracy for each approach.

Results in the above table evince that MO-EvoPruneDeepTL outperforms these pruning methods in most of the datasets. There are four datasets in which, without any doubt, MO-EvoPruneDeepTL achieves a better performance than pruning methods. For the SRSMAS dataset, *weight* is slightly better than MO-EvoPruneDeepTL in the case of the lowest percentage of active weights. This difference might be enough to state that SRSMAS performs better than MO-EvoPruneDeepTL. Nonetheless, the median case shows that, when a minimal number of neurons/weights are active, our proposal outperforms *weight* in this dataset.

A special case is noted in the results for the RPS dataset, which is the easiest one in terms of modeling difficulty. Results expose this fact because, when the approach is to eliminate a whole group of connections represented by the neurons, MO-EvoPruneDeepTL achieves a better performance in both cases. In fact, the greater the number of neurons/connections to be active is, the better both models will perform. However, if the strategy is to eliminate single connections as implemented by the *weight* strategy, it does not imply removing the whole set of connections of the neuron. In this case, this method may perform better than MO-EvoPruneDeepTL. The fact that RPS is the simplest dataset is reflected in the fact that the same accuracy value can be achieved by several desired pruning configurations. Based on this observation, it

Table 4: Comparison of MO-EvoPruneDeepTL against competitive pruning methods of the literature in terms of accuracy given a fixed value of target percentage of pruning weights.

Dataset	Percentage of remaining weights	`weight`	`neuron`	MO-EvoPruneDeepTL
CATARACT	3.3%	0.380	0.248	**0.694**
	0.09%	0.182	0.165	**0.504**
LEAVES	10.80%	0.637	0.700	**0.906**
	2.90%	0.462	0.450	**0.515**
PAINTING	15.5%	0.747	0.524	**0.935**
	0.09%	0.333	0.107	**0.429**
PLANTS	11.1%	0.212	0.072	**0.373**
	0.25%	0.043	0.034	**0.106**
RPS	5.0%	**0.943**	0.900	0.894
	0.24%	**0.943**	0.333	0.484
SRSMAS	8.79%	0.454	0.408	**0.782**
	0.10%	**0.161**	0.079	0.145

can be concluded that removing connections is a valid pruning method especially when complemented with other techniques such as evolutionary algorithms. In this case, there is potential for improvement in extreme, intermediate or general cases, as shown in the Pareto front estimations reported in these results.

Results attained by MO-EvoPruneDeepTL at the median percentage of pruning neurons are remarkable, since it corresponds to the center of the distribution of complexity values in the Pareto front estimated by the technique. In detail, all cases report a minimum of approximately 70% of pruned weights in the worst case. In the best case, almost the entire network is pruned, which corresponds to the lowest complexity value in the estimated front. The higher the percentage of pruned neurons is, the more difficult is to achieve a model with good accuracy levels, since a minimal amount of neurons/weights is needed to achieve them. This is exposed in most considered cases, in which the median value achieves a better performance. Taking a closer look at the case with lowest percentage of remaining weights, which can be deemed a more complex case, the performance of the models degrades, which is one of the lessons learned from the inspection of the Pareto fronts made in this section. However, MO-EvoPruneDeepTL is able to outperform the rest of pruning methods with models that do not surpass 3% of active neurons, except in the case of SRSMAS, whose performance is practically the same.

The Pareto fronts shown in this section have allowed us to obtain valuable information on the different executions of MO-EvoPruneDeepTL. The configuration of MO-EvoPruneDeepTL has allowed us to obtain a fairly diverse set of solutions, with competitive solutions at the extreme values of the different objectives of the study. A second conclusion drawn from this analysis is the existence of relationship or direct Pareto both the complexity of the network and its performance and the complexity and robustness, but it does not appear to exist between the performance and the robustness. Finally, a third conclusion has been drawn from the comparison against other pruning methods: in general, MO-EvoPruneDeepTL is able to outperform such methods for both intermediate and extreme pruning values, whereas a minimum percentage of neurons is required to produce high-quality pruned models.

5.2. Answering RQ2: Remarkable pruning patterns in the Pareto fronts of MO-EvoPruneDeepTL

This RQ aims to analyze if there are certain pruning patterns, along the different trained networks, that allows detecting important regions in the input images to the pruned networks.

In order to answer this RQ, we must discriminate relevant neurons that appear in most of the pruning patterns in the Pareto fronts produced by MO-EvoPruneDeepTL. In doing so, we resort to a XAI technique

called GradCAM (Selvaraju et al., 2017, October), which permits to localize the regions within the image that are responsible for a class prediction. Thanks to GradCAM, we can go backwards from the neurons of the solutions and highlight these key pixel regions. For each dataset, we depict several query images, and remark the 10 most relevant neurons as per GradCAM and their distribution among the three objectives. In the following figures, the central sections are relevance heatmaps obtained by GradCAM, remarking the most influential zones of the input images as warmer colors. In addition to the heatmap, we also present two more plots. The first one, in the left top, show as a bar diagram the index of the 10 most relevant neurons which appear as active in most of the solutions of the Pareto front, with their relative frequency. The second one, at the right, shows the distribution of the objective's values for these representative neurons. In this chart, a boxplot is shown for each objective and neuron: from left to right, accuracy, percentage of active neurons and AUROC, respectively. The border of the heatmaps and the color of the bars in the figures are related, so that the reader can match each heatmap to the corresponding neuron and frequency of appearance in the Pareto.

Figure 5 shows the previous information for the CATARACT dataset. In the first one, the barplot, we can see that the least important neuron achieves a 60% of frequency in the Pareto front, i.e., it appears in the 60% of solutions meanwhile the best one has a frequency rate of more than the 80%. The second figure, the boxplot, shows the distribution of the objectives for solutions which have these relevant neurons. These results show that low complexity is presented in these neurons and high accuracy and AUROC. Lastly, we see the heatmaps for this dataset. For the shown images, we can see how these pruning patterns that MO-EvoPruneDeepTL achieves during its evolutionary process. These patterns let us recognize how the network dictate the class for each image thanks to these ten most important neurons.

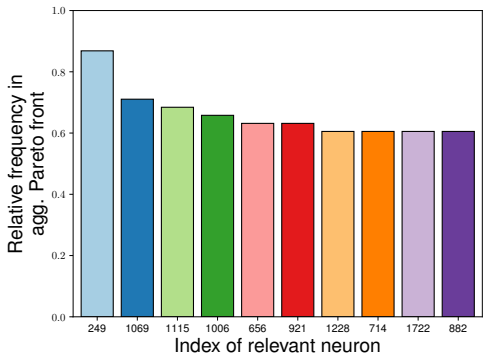
The next figure, Figure 6 shows the results for the RPS dataset. The bar graph shows that these neurons achieve an appearance in more than the 80% of solutions of the Pareto and the boxplot confirm that the solutions in which these neurons are presented achieve, in most cases, less than 10% of active neurons, accuracies near 90% and AUROC around an 80%. The examples images shown in the heatmaps present the effect of these important neurons. As we have previously noted, the keys to recognize the images are position of the fingers and even the separation among them, as warmer color are presented for them.

The next dataset is PAINTING. Figure 7 shows the set of graphics for this dataset. The relevant neurons for PAINTING achieve a minimum percentage of appearance of 70% among all the solutions in the Pareto front. There is a significant difference between the first relevant neuron and the rest in terms of appearance. These solutions present almost a 20% of active neurons, but also high performance both in accuracy and AUROC, between 90 and 100%. This indicates the great level of uniformity in the robustness for this dataset. These neurons help us to analyze the images of this dataset. The third image presents a woman and, taking a deep look into the heatmaps, we see that the network recognizes the face, and then the outer parts, like the arms and the hair. Another interesting image is the fifth one. Our network is able to recognize the chest and also the arms and the rest of the body extremities.

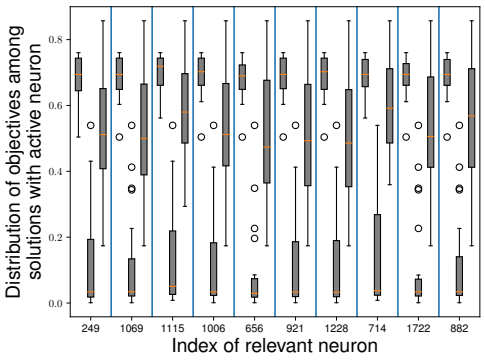
We continue our analysis of the obtained pruning patterns of MO-EvoPruneDeepTL with the PLANTS dataset, shown in Figure 8. The most important neurons have an appearance rate between 60 and 80% in all the solutions of the Pareto front. Their distribution of objective report us a very low complexity of the network, near the 10% in average, with a good result for this dataset both in accuracy and AUROC. This dataset contains images of leaves and plants of fruit and vegetables and, for that reason, our network focus in the recognition of the shape of these leaves, as it is shown in the three bottom images of the figure.

We continue with this analysis with the LEAVES dataset. In this case, Figure 9 shows the three graphics for this dataset. The first one, which is related with the relevance of the neurons, exhibit two neurons which appear in all the solutions of the Pareto front and another two which present almost a 100% of appearance. For those neurons, the boxplot chart report us similar distributions because, in all the cases, the remaining active neurons are kept low and the accuracy and AUROC are high. Lastly, the images from this dataset show both diseased and healthy leaves. The achieved pruning patterns of MO-EvoPruneDeepTL are able to distinguish the healthy from the diseased leaves (last image versus the third one starting from the top), and then the type of the disease.

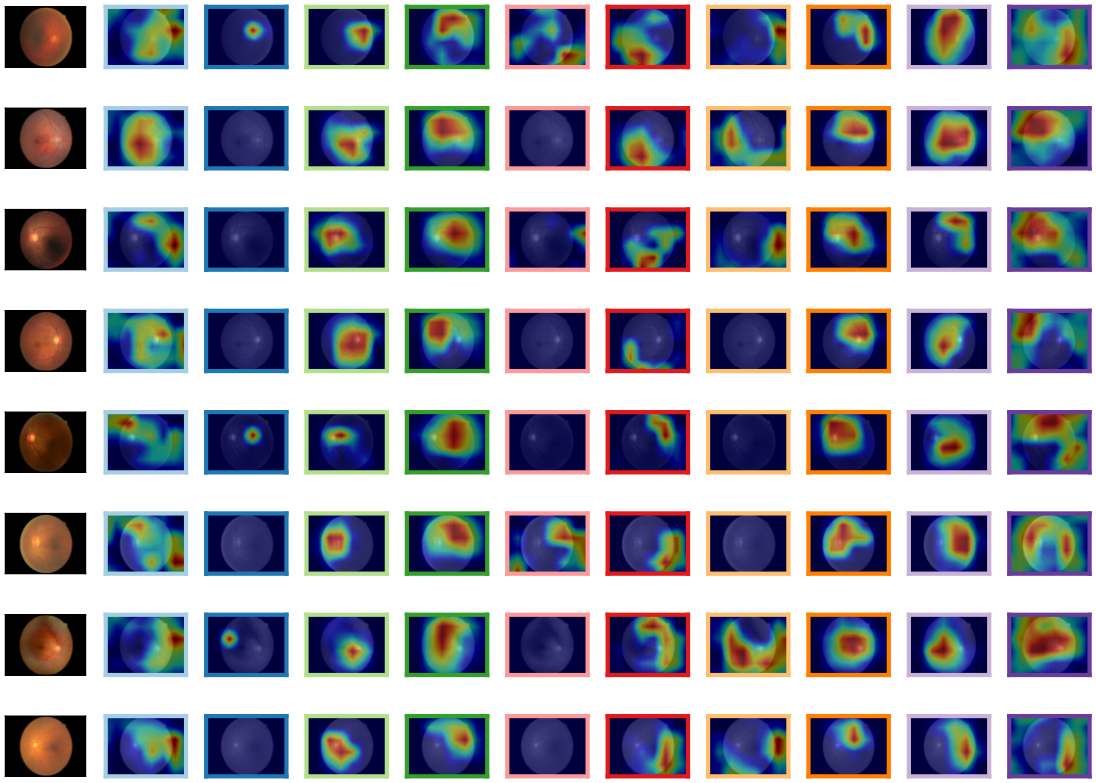
The last dataset is SRSMAS, whose charts are presented in Figure 10. The most relevant neurons obtain a minimum of 60% of appearance in all the solutions of the Pareto front, which has been a constant factor



(a) Bars of CATARACT

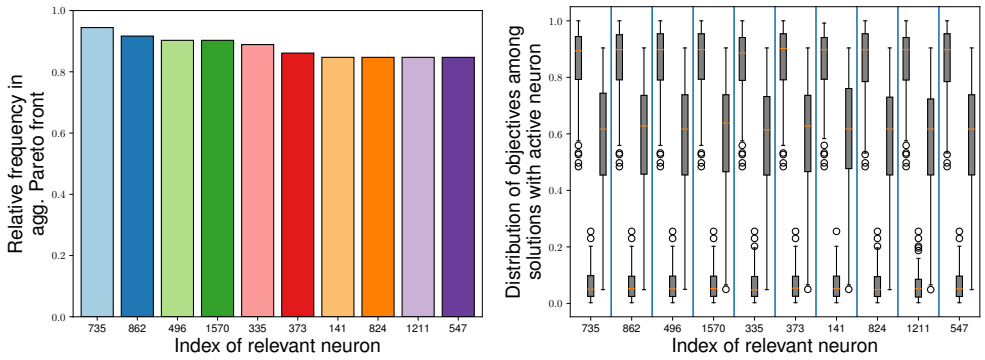


(b) Boxplots of CATARACT



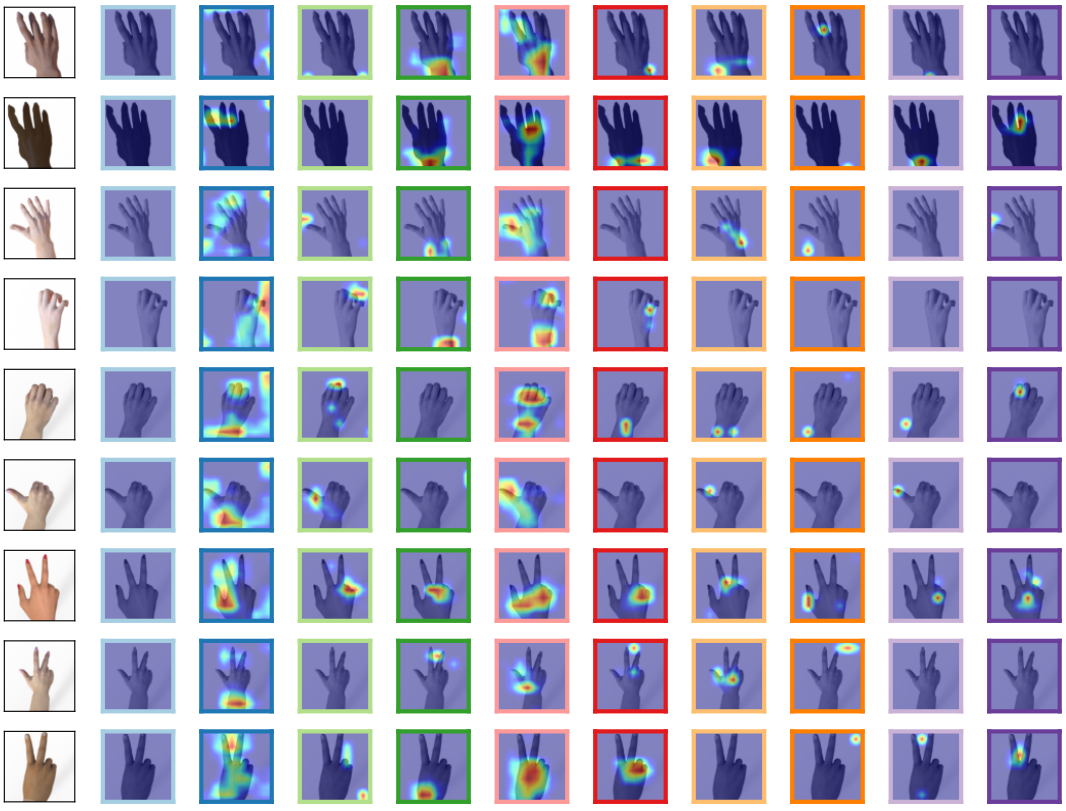
(c) Heatmaps of CATARACT

Figure 5: Bars, boxplots and heatmaps of CATARACT.



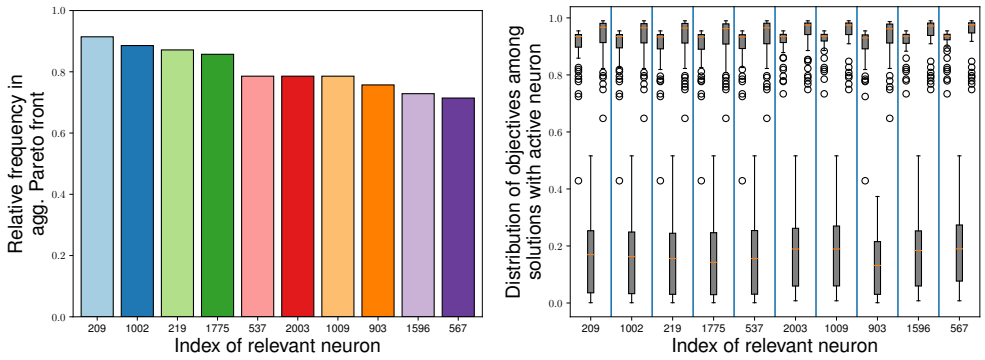
(a) Bars of RPS

(b) Boxplots of RPS



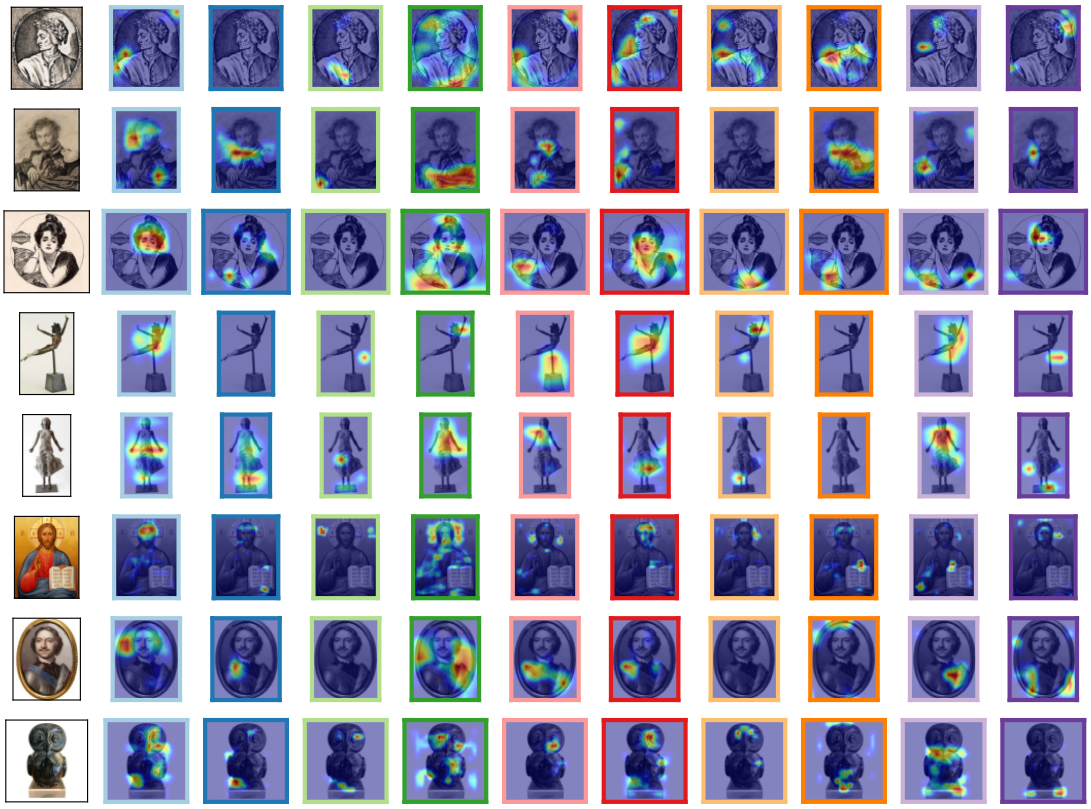
(c) Heatmaps of RPS

Figure 6: Bars, boxplots and heatmaps of RPS.



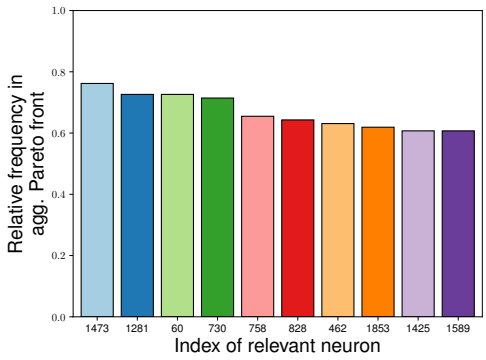
(a) Bars of PAINTING

(b) Boxplots of PAINTING

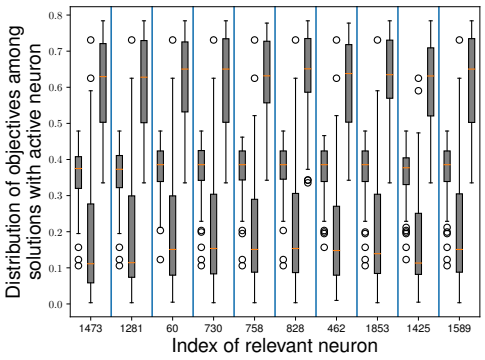


(c) Heatmaps of PAINTING

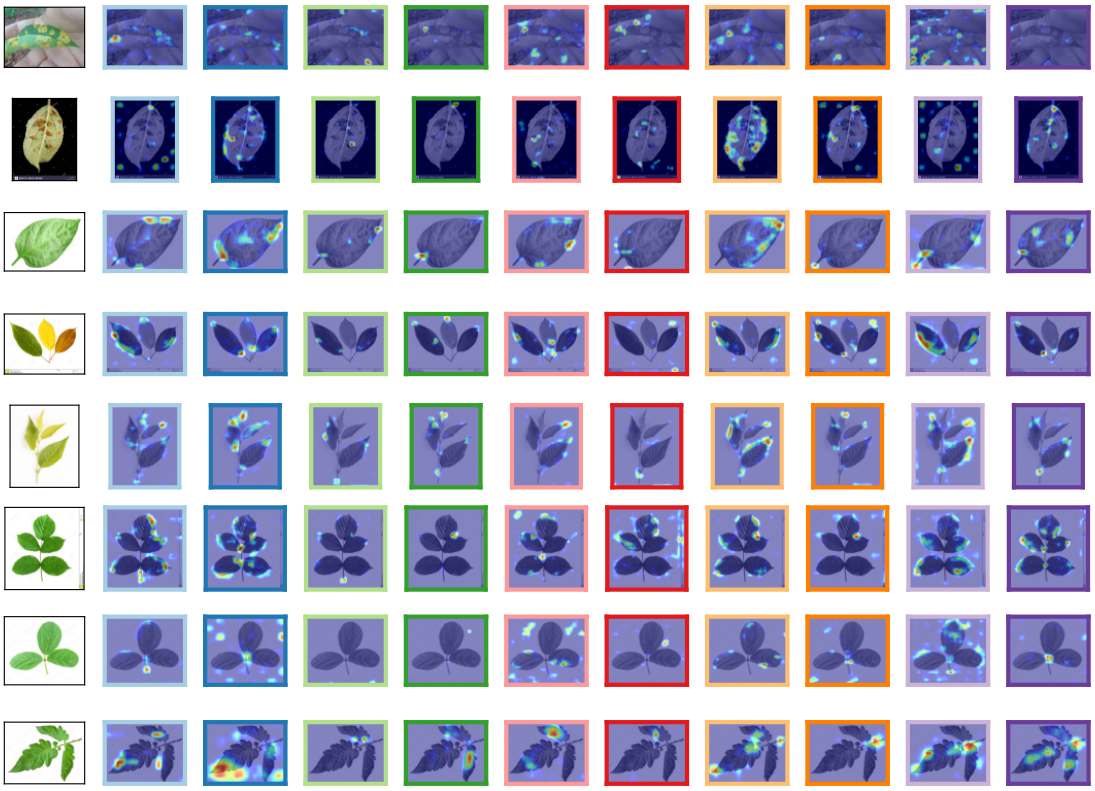
Figure 7: Bars, boxplots and heatmaps of PAINTING.



(a) Bars of PLANTS

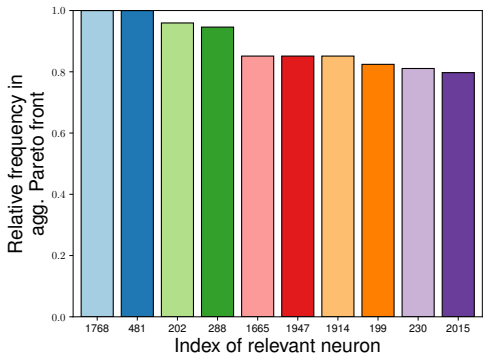


(b) Boxplots of PLANTS

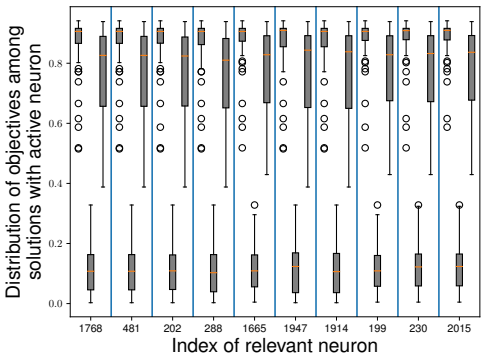


(c) Heatmaps of PLANTS

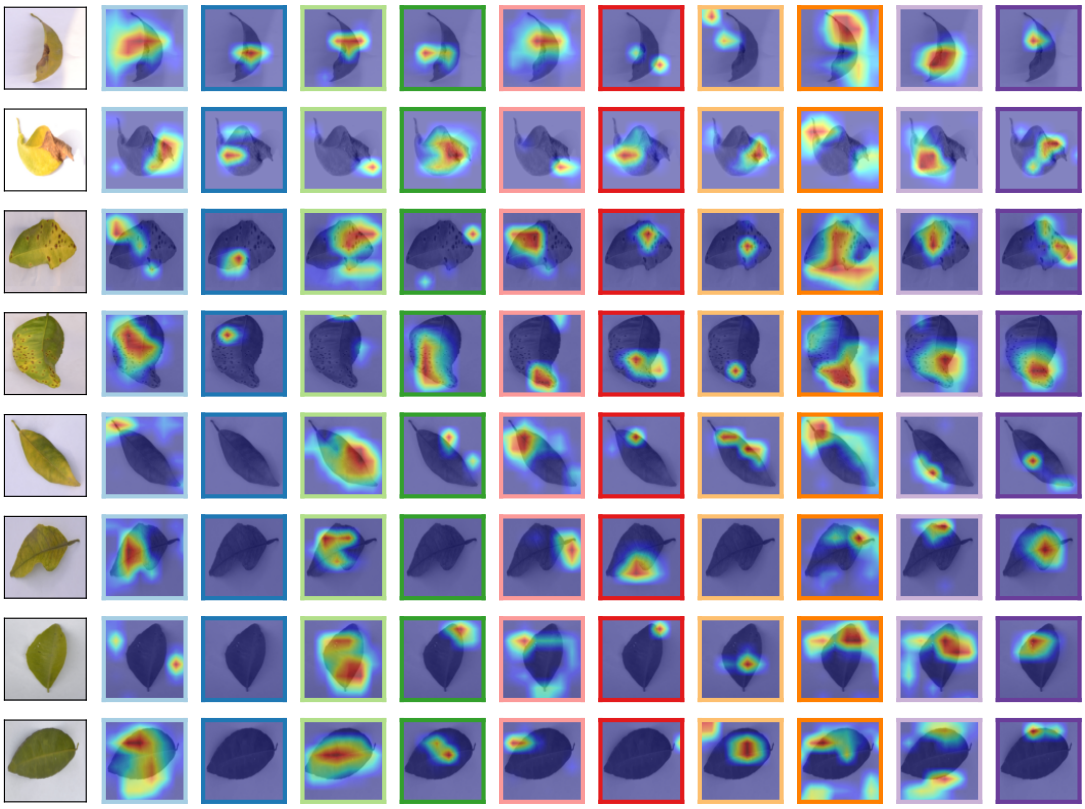
Figure 8: Bars, boxplots and heatmaps of PLANTS.



(a) Bars of LEAVES



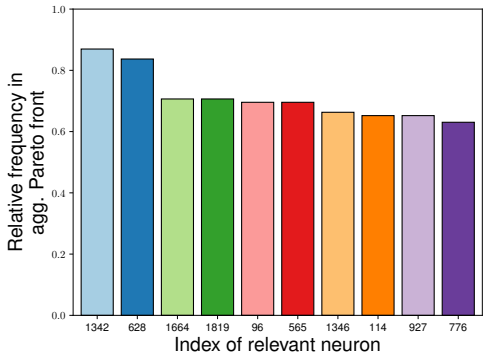
(b) Boxplots of LEAVES



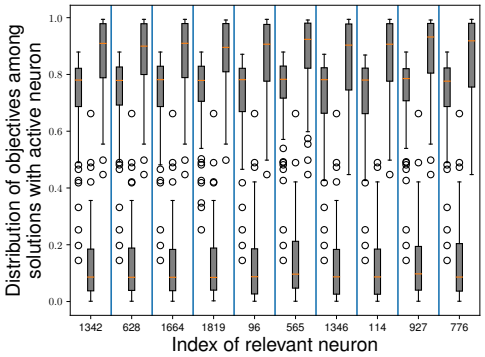
(c) Heatmaps of LEAVES

Figure 9: Bars, boxplots and heatmaps of LEAVES.

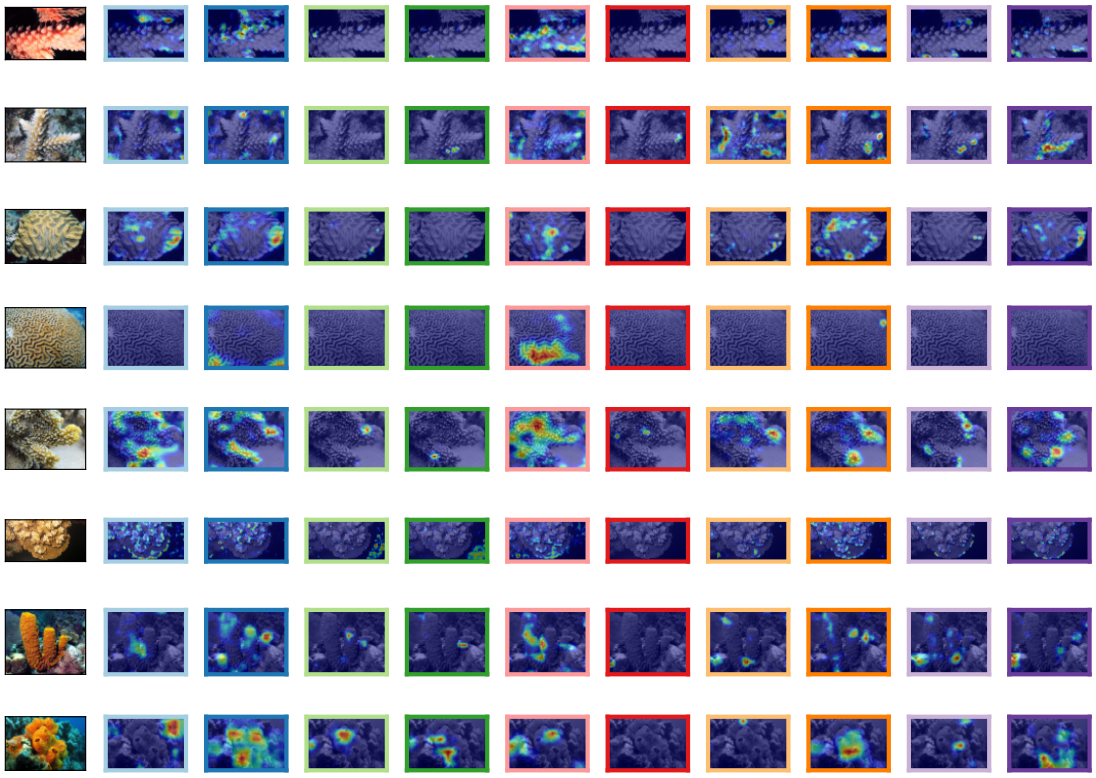
in all the datasets. Moreover, the distribution of the objectives for the solutions, in which these neurons are active, shares a common line: high values both performance and robustness and low complexity of the network. These neurons draw pruning patterns that identify the class for the input images. As an example, in the fourth image it is only necessary to recognize the silhouette of the coral reef, but in the fifth one, the network needs to understand how is the central part of the coral reef and then its extremities.



(a) Bars of SRSMAS



(b) Boxplots of SRSMAS



(c) Heatmaps of SRSMAS

Figure 10: Bars, boxplots and heatmaps of SRSMAS.

In these figures, we have seen several datasets in which the difference rate between the most important neurons is close (RPS and PLANTS), but there are other datasets in which this difference is up to 20% between the most and least relevant neurons. Moreover, the distribution of the objectives for each dataset

gives us good insights about the uniformity of the performance and robustness in most of the datasets.

The good work done by MO-EvoPruneDeepTL in training the models has made possible to achieve remarkable pruning patterns. These have helped us to decipher not only those neurons that have been key in the whole training and inference process, but also to locate in the input images those groups of influential pixels which have been important to decide the class of each of these images.

5.3. Answering RQ3: Quality of the models through ensemble modeling

This subsection is devised to formally answer to RQ3, which is to show if the ensemble modeling is able to improve the quality of the trained models by MO-EvoPruneDeepTL. An elementary key in this regard is model diversity, understood as the ability to generate and train models from a given dataset that are different to each other and that model differently the distribution underlying the dataset at hand. If MO-EvoPruneDeepTL is found to be capable of generating models in this way (as a byproduct of its multi-objective search), then an ensemble of such models can give rise to an improved performance and reduced risk of overfitting. For all of these reasons, ensemble modeling can achieve an improvement in terms of either accuracy and/or robustness with respect to the individual pruned models comprised in the Pareto front estimated by MO-EvoPruneDeepTL.

Having established the motivation for ensemble modeling, we will now describe its implementation. We depart from the two objectives to be maximized, namely, accuracy and robustness. The proposed ensemble strategy consists of collecting the models in the estimated Pareto front (containing the best solutions from the different runs performed) that fall within a statistical range of accuracy or AUROC (the robustness measure). Thus, the ensemble will fuse together those pruned models that fall within two given percentiles of the distribution of these metrics over the Pareto front of the three objectives. From these assembled models, their predictions for a given query are merged into one (by simple majority voting), and compared to the prediction of the best individual model in the ensemble.

The analysis of the ensemble behavior is done based on different percentile ranges of each of the accuracy and AUROC distributions, providing more precise information for each of these metrics. Next, we explain how such percentile ranges are chosen. We start with a first range (percentiles $(50\%, 60\%)$), and we increase the extremes of the interval by 5% in each iteration, giving us 8 quartile intervals for both metrics. Models in the Pareto front whose objective values fall within each of these percentile ranges are included in the ensemble. For example, the interval $(75\%, 85\%)$ will contain those models in the estimated Pareto front whose accuracy objective is within this range given the distribution of the accuracy objective computed over the whole Pareto front estimation (a similar example can be given for the AUROC score). These percentiles are defined as (Q_{min}, Q_{max}), where $Q_{min} = 50\%, 55\%, \ldots, 85\%$, and $Q_{max} = 60\%, 65\%, \ldots, 95\%$. With this division, we have the following intervals $(50\%, 60\%), (55\%, 65\%), \ldots, (85\%, 95\%)$.

In this study, we have selected the CATARACT, PAINTING and RPS datasets for the experimental tests performed to examine the behavior of ensemble modeling. Two different plots are depicted for each dataset, one for each metric (accuracy and AUROC). In each of these plots, three symbols appear in the form of a rectangle, a square and a star. The rectangle shows the distribution of accuracy/AUROC values for the models in the percentile range at hand. The square symbolizes the best result for that measure. Lastly, the star indicates the accuracy/AUROC of the ensemble.With this explanation, we can interpret the two graphs that result from making the ensemble. The first one is related to the accuracy of the network. Figure 11 shows this graph. It presents three graphs sorted alphabetically by dataset. The first one corresponds to CATARACT, the second to RPS and the third to PAINTING. Each of them shows, for each interval of quantile the distribution of individual accuracies, the maximum of the distribution and the accuracy of the ensemble.

The overall performance in the three cases is positive since the diversity of the models allows us to find new models that improve the accuracy for each quantile interval, except in the case of RPS where we only have one model in the interval $(60\%, 70\%)$. In the RPS case, we have models near the 90% of accuracy and the ensemble produces a new model with almost 96% of it, which is a great result. Moreover, models with higher accuracy (96% or more) achieve close to 100% of accuracy. For RPS (the chart of the right), most of the ensemble models get a 95% of accuracy, meanwhile their individual models are present a lower value in accuracy. As a result, these charts show the benefits of the ensemble modeling for the accuracy objective.

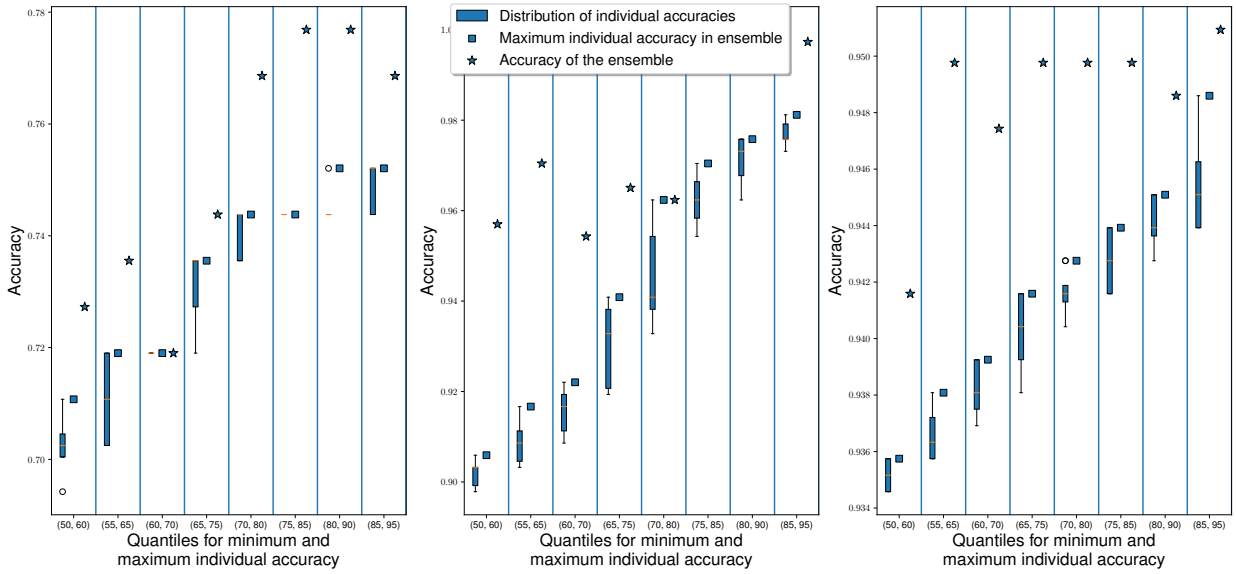


Figure 11: Ensemble modeling of the models trained by MO-EvoPruneDeepTL in terms of accuracy. Left: CATARACT dataset. Middle: RPS dataset. Right: PAINTING dataset.

The second part of this section consists of replicating the previous experiment, but for the case of OoD detection in order to check if the AUROC improves when ensemble modeling occurs. In such a case, we will be able to confirm that the new model detects less OoD sample as InD, which makes an improvement in the associated metric.

The interpretation of the set of charts is the same as in the previous case. We have made the ensemble with the models for each interval. The same characteristics are presented in Fig. 12. It is shown the distribution of individual AUROC values, its maximum and then, marked with a star, the AUROC of the ensemble. The CATARACT case shows an improvement in the AUROC in all the cases but one. For the RPS case (middle chart), the case of $(85\%, 95\%)$ achieves almost a 95% of AUROC, meanwhile the individual values get a maximum of 87%. The PAINTING dataset also presents outstanding results. Its minimum AUROC for all the intervals of the ensemble is more than 98.5% and the least value is of individual models is less than 97%. The results obtained from the graph are similar to those obtained for the case of accuracy, since they improve on the individual results in the vast majority of the intervals.

In this section, we have conducted two experiments which involve the ensemble modeling of the trained models by MO-EvoPruneDeepTL. The ensemble has been done taking into account the performance of the network and the robustness and we have given the liberty to choose the interval of values for each measure. The results drawn from these graphics show that both of the objectives have been improved. Performing a MO search not only provides the user with a wide range of models that balance between the three stated objectives, but it also achieves more diversity among the models in order to ensemble them and achieve even higher performance and robustness.

6. Conclusions

This paper has introduced MO-EvoPruneDeepTL, a MONAS model that evolves sparse layers of a DL model which has been instantiated using the TL paradigm. MO-EvoPruneDeepTL uses a MOEA, which evolves these sparse layers, in order to obtain adapted, pruned layers to the problem at hand and making decisions about the neurons that need to be active or inactive.

MO-EvoPruneDeepTL is a model that evolves the extracted features from the pre-trained network in order to train the last layers to tackle the considered problem. Our results draw two conclusions from

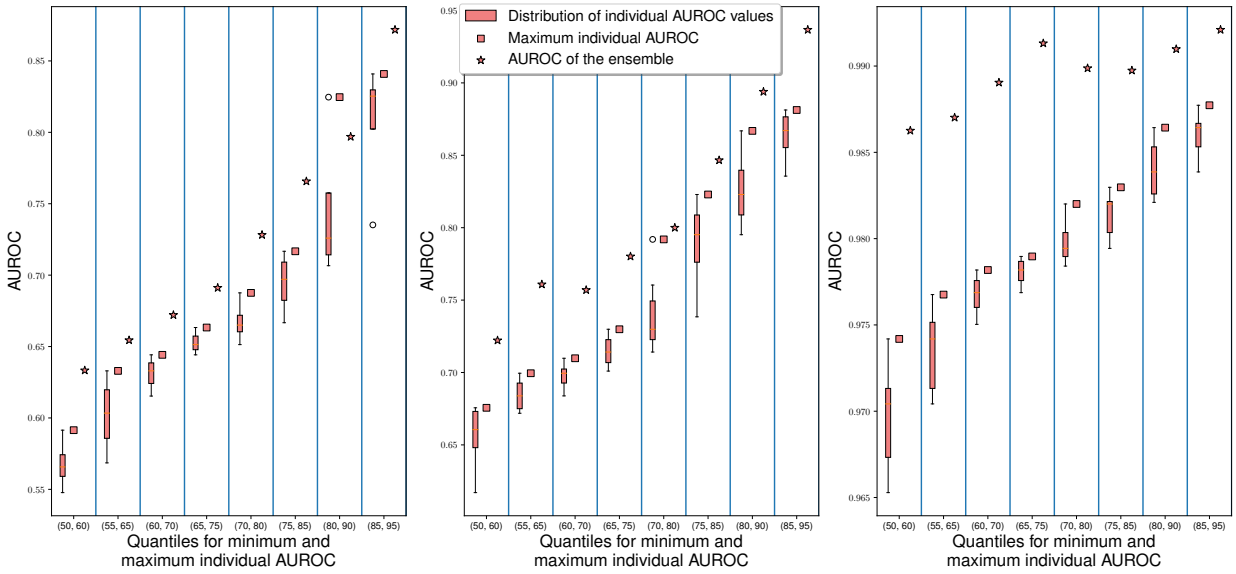


Figure 12: Ensemble modeling of the models trained by MO-EvoPruneDeepTL in terms of OoD detection. Left: CATARACT dataset. Middle: RPS dataset. Right: PAINTING dataset.

the Pareto fronts: there exists a great diversity in the solutions and they also establish promising values for the objectives in their extremes values. Moreover, the projections for each objective shed light on the existence of a direct relationships between the complexity of the network and each of the other two objectives (performance and robustness), whereas there is no direct relationship between the latter two. This work falls within the umbrella of OWL because the evolved models are asked about new data, which is the OoD datasets. Moreover, OWL is related with GPAI and, particularly, in this manuscript, the experiments have shown the capability of AI generating AI as the MOEA has learnt from the trained DL models.

The trained models of MO-EvoPruneDeepTL lead to several pruning patterns in which there exist neurons that appear in most of the best solutions of the Pareto front. These patterns help us to recognize the key group of regions of the input images that our models consider the most important ones when assigning the class to the input image at inference time.

The diversity of the models of MO-EvoPruneDeepTL has shown that ensemble modeling is able to increase the overall performance, both in performance of the network and robustness, in most of the quantiles for minimum and maximum considered objective values.

The evolved trained models have shown a great performance with a minimum number of active neurons, but it is also shown the great contribution of the robustness for these models, as each DL model is tested with data that it has not previously seen. Moreover, the objectives of the MOEA have been the performance, complexity and robustness, but other alternatives can be formulated as objectives such as the latency or energy used of the GPU in the inference of the pruned model or the epistemic uncertainty level.

An ablation study is also in our agenda for future research, aiming to discern which algorithmic steps are more relevant for the search convergence of the solver when tackling the multi-objective problem at hand. We envision that the results of this ablation study can illuminate the design of new operators and more effective search strategies than the ones utilized in this work. Moreover, we will investigate the influence of different robustness measures on the Pareto front estimations produced by MO-EvoPruneDeepTL.

Acknowledgments

F. Herrera, D. Molina and J. Poyatos are supported by the Andalusian Excellence project P18-FR-4961, the infrastructure project with reference EQC2018-005084-P and the R&D and Innovation project with

reference PID2020-119478GB-I00 granted by the Spain's Ministry of Science and Innovation and European Regional Development Fund (ERDF). A. Martinez-Seras and J. Del Ser would like to thank the Basque Government for the funding support received through the EMAITEK and ELKARTEK programs, as well as the Consolidated Research Group MATHMODE (IT1456-22) granted by the Department of Education of this institution.

References

Assunção, F., Lourenço, N., Machado, P., & Ribeiro, B. (2019). Denser: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines*, *20*, 5–35. https://doi.org/10.1007/s10710-018-9339-y.

Back, T., Fogel, D. B., & Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*. (1st ed.). IOP Publishing Ltd.

Back, T., & Schwefel, H.-P. (1996, May). Evolutionary computation: an overview. In *Proceedings of IEEE International Conference on Evolutionary Computation, Padua, Italy, 1996*. https://doi.org/10.1109/ICEC.1996.542329.

Baldeon Calisto, M., & Lai-Yuen, S. K. (2020). AdaEn-Net: An ensemble of adaptive 2D–3D Fully Convolutional Networks for medical image segmentation. *Neural Networks*, *126*, 76–94. https://doi.org/10.1016/j.neunet.2020.03.007.

Baldeon-Calisto, M., & Lai-Yuen, S. K. (2020). AdaResU-Net: Multiobjective adaptive convolutional neural network for medical image segmentation. *Neurocomputing*, *392*, 325–340. https://doi.org/10.1016/j.neucom.2019.01.110.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012.

Calisto, M. B., & Lai-Yuen, S. (2020). Neural Architecture Search with an Efficient Multiobjective Evolutionary Framework. *arXiv preprint arXiv:2011.04463*, . https://arxiv.org/abs/2011.04463.

Chitty-Venkata, K. T., & Somani, A. K. (2022). Neural Architecture Search Survey: A Hardware Perspective. *ACM Computing Surveys*, *55*, 1–36. https://doi.org/10.1145/3524500.

[dataset]Sungjoon Choi (2020). Cataract Dataset. Retrieved from https://www.kaggle.com/jr2ngb/cataractdataset. Accessed September 10, 2020.

Clune, J. (2019). AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*, . https://arxiv.org/abs/1905.10985.

Deb, K. (2011). *Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction*. Springer London.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197. https://doi.org/10.1109/4235.996017.

Dufourq, E., & Bassett, B. A. (2017, November). EDEN: Evolutionary deep networks for efficient machine learning. In *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech), Bloemfontein, South Africa, 2017* (pp. 110–115). https://doi.org/10.1109/RoboMech.2017.8261132.

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *The Journal of Machine Learning Research*, *20*, 1997–2017. http://jmlr.org/papers/v20/18-598.html.

Elsken, T., Metzen, J. H., & Hutter, F. (2019, May). Efficient Multi-Objective Neural Architecture Search via Lamarckian Evolution. In *7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 2019*. https://doi.org/10.48550/arXiv.1804.09081.

[dataset] Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A., & Herrera, F. (2019). Coral species identification with texture or structure images using a two-level classifier based on Convolutional Neural Networks. *Knowledge-Based Systems*, *184*, 104891. https://doi.org/10.1016/j.knosys.2019.104891.

Han, S., Pool, J., Tran, J., & Dally, W. (2015, December). Learning both Weights and Connections for Efficient Neural Network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal Canada, 2015*. https://papers.nips.cc/paper_files/paper/2015/hash/ae0eb3eed39d2bcef4622b2499a05fe6-Abstract.html.

Hendrycks, D., & Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, . https://doi.org/10.48550/arXiv.1610.02136.

Hendrycks, D., Mazeika, M., & Dietterich, T. (2019, May). Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations, New Orleans, USA, 2019*. https://openreview.net/forum?id=HyxCxhRcY7.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in Deep Learning: Pruning and Growth for Efficient Inference and Training in Neural Networks. *Journal of Machine Learning Research*, *22*, 1–124.

ISO (2021a). *Artificial Intelligence (AI) — Assessment of the robustness of neural networks — Part 1: Overview*. Technical Report ISO/IEC JTC 1/SC 42 Artificial intelligence (24029-1:2021).

ISO (2021b). *Artificial intelligence (AI) — Assessment of the robustness of neural networks — Part 2: Methodology for the use of formal methods*. Technical Report ISO/IEC JTC 1/SC 42 Artificial intelligence (24029-2).

Khan, S., Islam, N., Jan, Z., Din, I. U., & Rodrigues, J. J. C. (2019). A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognition Letters*, *125*, 1–6. https://doi.org/10.1016/j.patrec.2019.03.022.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM*, *60*, 84–90. https://doi.org/10.1145/3065386.

Lee, K., Lee, K., Lee, H., & Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial

attacks. *Advances in Neural Information Processing Systems*, *31*. https://proceedings.neurips.cc/paper/2018/file/abdeb6f575ac5c6676b747bca8d09cc2-Paper.pdf.

Liang, S., Li, Y., & Srikant, R. (2018, Aprila). Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, Canada, 2018*. https://doi.org/10.48550/arXiv.1706.02690.

Liang, S., Li, Y., & Srikant, R. (2018, Aprilb). Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. In *International Conference on Learning Representations, Vancouver, Canada, 2018*. https://openreview.net/forum?id=H1VGkIxRZ.

Lin, Z., Roy, S. D., & Li, Y. (2021, June). MOOD: Multi-level Out-of-distribution Detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, Tennessee ,2021*. IEEE. https://doi.org/10.1109/cvpr46437.2021.01506.

Liu, W., Wang, X., Owens, J., & Li, Y. (2020). Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, *33*, 21464–21475. https://proceedings.neurips.cc/paper/2020/hash/f5496252609c43eb8a3d147ab9b9c006-Abstract.html.

Loni, M., Sinaei, S., Zoljodi, A., Daneshtalab, M., & Sjödin, M. (2020). DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems. *Microprocessors and Microsystems*, *73*, 102989. https://doi.org/10.1016/j.micpro.2020.102989.

Lu, Z., Cheng, R., Huang, S., Zhang, H., Qiu, C., & Yang, F. (2022a). Surrogate-assisted Multi-objective Neural Architecture Search for Real-time Semantic Segmentation. *arXiv preprint arXiv:2208.06820*, . https://arxiv.org/abs/2208.06820.

Lu, Z., Cheng, R., Jin, Y., Tan, K. C., & Deb, K. (2022b). Neural Architecture Search as Multiobjective Optimization Benchmarks: Problem Formulation and Performance Assessment. *arXiv preprint arXiv:2208.04321*, . https://doi.org/10.48550/arXiv.2208.04321.

Lu, Z., Deb, K., Goodman, E., Banzhaf, W., & Boddeti, V. N. (2020, August). NSGANetV2: Evolutionary Multi-Objective Surrogate-Assisted Neural Architecture Search. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, Proceedings, Part I*. https://doi.org/10.1007/978-3-030-58452-8_3.

Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., & Banzhaf, W. (2019, July). NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 2019*. https://doi.org/10.24963/ijcai.2020/659.

Lu, Z., Whalen, I., Dhebar, Y., Deb, K., Goodman, E. D., Banzhaf, W., & Boddeti, V. N. (2021). Multiobjective Evolutionary Design of Deep Convolutional Neural Networks for Image Classification. *IEEE Transactions on Evolutionary Computation*, *25*, 277–291. https://doi.org/10.1109/TEVC.2020.3024708.

Martín, A., Lara-Cabrera, R., Fuentes-Hurtado, F., Naranjo, V., & Camacho, D. (2018). EvoDeep: a new evolutionary approach for automatic deep neural networks parametrisation. *Journal of Parallel and Distributed Computing*, *117*, 180–191. https://doi.org/10.1016/j.jpdc.2017.09.006.

Martinez, A. D., Del Ser, J., Villar-Rodriguez, E., Osaba, E., Poyatos, J., Tabik, S., Molina, D., & Herrera, F. (2021). Lights and shadows in Evolutionary Deep Learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges. *Information Fusion*, *67*, 161–194. https://doi.org/10.1016/j.inffus.2020.10.014.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N. et al. (2019). Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing* (pp. 293–312). Academic Press. https://doi.org/10.1016/B978-0-12-815480-9.00015-3.

[dataset]Laurence Moroney (2019). Rock, Paper, Scissors Dataset. Retrieved from http://www.laurencemoroney.com/rock-paper-scissors-dataset/. Accessed September 10, 2020.

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*, 1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Parmar, J., Chouhan, S. S., Raychoudhury, V., & Rathore, S. S. (2022). Open-World Machine Learning: Applications, Challenges, and Opportunities. *ACM Comput. Surv.*, (pp. 1–36). https://doi.org/10.1145/3561381.

Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning, Stockholm, Sweden, 2018*. http://proceedings.mlr.press/v80/pham18a.html.

Poyatos, J., Molina, D., Martinez, A. D., Del Ser, J., & Herrera, F. (2023). EvoPruneDeepTL: An evolutionary pruning model for transfer learning based deep neural networks. *Neural Networks*, *158*, 59–82. https://doi.org/10.1016/j.neunet.2022.10.011.

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019, January). Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 2019*. https://doi.org/10.1609/aaai.v33i01.33014780.

Real, E., Liang, C., So, D., & Le, Q. (2020, July). AutoML-zero: evolving machine learning algorithms from scratch. In *International Conference on Machine Learning, 2020*. https://doi.org/10.48550/arXiv.2003.03384.

[dataset]Virtual Russian Museum (2018). Art Images: Drawing/Painting/Sculptures/Engravings. Retrieved from https://www.kaggle.com/thedownhill/art-images-drawings-painting-sculpture-engraving, Accessed September 10, 2020.

Salehi, M., Mirzaei, H., Hendrycks, D., Li, Y., Rohban, M. H., & Sabokrou, M. (2021). A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv preprint arXiv:2110.14051*, . https://doi.org/10.48550/arXiv.2110.14051.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017, October). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV), Venezia, Italy, 2017*. https://doi.org/10.1109/ICCV.2017.74.

[dataset] Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S., & Batra, N. (2020, May). PlantDoc: A Dataset for Visual Plant Disease Detection. In *7th ACM IKDD CoDS and 25th COMAD, Hyderabad, India, 2020*. https://doi.org/10.1145/3371158.3371196.

Srinivas, S., & Babu, R. V. (2015, September). Data-free Parameter Pruning for Deep Neural Networks. In *Proceedings of the British Machine Vision Conference (BMVC), Swansea, UK, 2015*. https://dblp.org/rec/journals/corr/SrinivasB15.bib.

Stanley, K., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*, 99–127. https://doi.org/10.1162/106365602320169811.

Suganuma, M., Kobayashi, M., Shirakawa, S., & Nagao, T. (2020). Evolution of Deep Convolutional Neural Networks Using Cartesian Genetic Programming. *Evolutionary Computation*, *28*, 141–163. https://doi.org/10.1162/evco_a_00253.

[dataset]Hafiz Tayyab Rauf, Saleem, B. A., Lali, M. I. U., Khan, M. A., Sharif, M., & Bukhari, S. A. C. (2019). A citrus fruits and leaves dataset for detection and classification of citrus diseases through machine learning. *Data in Brief*, *26*, Article 104340. https://doi.org/10.1016/j.dib.2019.104340.

Trivedi, A., Srivastava, S., Mishra, A., Shukla, A., & Tiwari, R. (2018). Hybrid evolutionary approach for Devanagari handwritten numeral recognition using Convolutional Neural Network. *Procedia Computer Science*, *125*, 525–532. https://doi.org/10.1016/j.procs.2017.12.068.

Wang, S., Lin, P., Hu, R., Wang, H., He, J., Huang, Q., & Chang, S. (2019). Acceleration of LSTM With Structured Pruning Method on FPGA. *IEEE Access*, *7*, 62930–62937. 10.1109/ACCESS.2019.2917312.

Wang, S., Liu, J., & Jin, Y. (2021). A Computationally Efficient Evolutionary Algorithm for Multiobjective Network Robustness Optimization. *IEEE Transactions on Evolutionary Computation*, *25*, 419–432. https://doi.org/10.1109/TEVC.2020.3048174.

Wang, Z., Li, F., Shi, G., Xie, X., & Wang, F. (2020). Network pruning using sparse learning and genetic algorithm. *Neurocomputing*, *404*, 247–256. https://doi.org/10.1016/j.neucom.2020.03.082.

Wei, H., Lee, F., Hu, C., & Chen, Q. (2022). MOO-DNAS: Efficient Neural Network Design via Differentiable Architecture Search Based on Multi-Objective Optimization. *IEEE Access*, *10*, 14195–14207. https://doi.org/10.1109/ACCESS.2022.3148323.

Yang, J., Zhou, K., Li, Y., & Liu, Z. (2021). Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, . https://doi.org/10.48550/arxiv.2110.11334.

Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., & Xu, C. (2020, June). CARS: Continuous Evolution for Efficient Neural Architecture Search. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, June, 2020*. https://doi.org/10.1109/CVPR42600.2020.00190.

Zhou, Z.-H. (2022). Open-environment machine learning. *National Science Review*, *9*, 1–11. https://doi.org/10.1093/nsr/nwac123.

Zoph, B., & Le, Q. V. (2016). Neural Architecture Search with Reinforcement Learning. *arXiv preprint arXiv:1611.01578*, . https://arxiv.org/abs/1611.01578.

Chapter III

El papel de la Computación Evolutiva en los sistemas de IA de propósito general

1 Introducción

En este capítulo se abordará el cuarto objetivo de la tesis. Este trabajo aborda la creciente demanda de modelos adaptables capaces de hacer frente a un conjunto diverso de tareas de aprendizaje, superando las limitaciones de los sistemas diseñados para ocuparse de una sola tarea. La reciente aparición de los Sistemas de Inteligencia Artificial de Propósito General (*General-Purpose Artificial Intelligence Systems*- GPAIS) plantea retos tanto de configuración como de adaptabilidad de modelos a escalas de complejidad mucho mayores que el diseño óptimo de los modelos tradicionales de Aprendizaje Automático (*Machine Learning - ML)*. La Computación Evolutiva ha sido una herramienta útil tanto para el diseño como para la optimización de modelos de ML, dotándolos de la capacidad de configurarse y/o adaptarse a la tarea considerada. Por ello, su aplicación a GPAIS puede considerarse como una elección natural.

El objetivo de este trabajo es analizar el papel de la Computación Evolutiva en el campo de los GPAIS, explorando el uso de Algoritmos Evolutivos (AEs) para su diseño o mejora. También relacionamos las propiedades de los GPAIS con las áreas del ML en las que los AEs han tenido una notable contribución, destacando los hitos recientes de los AEs para los GPAIS. Además, discutimos los retos de aprovechar los beneficios de los AEs para los GPAIS, presentando diferentes estrategias tanto para el diseño como para la mejora de los GPAIS con AEs, cubriendo áreas tangenciales, identificando nichos de investigación y esbozando potenciales líneas de investigación para los AEs y los GPAIS.

Este trabajo se estructura de la siguiente manera: la Sección 1.1 introduce y motiva las bases con las que se trabajará en el resto de secciones. Tras ello, en la Sección 2, se proporciona la base necesaria para comprender el trabajo realizado. En la Sección 3, se examinan las diferentes opciones de uso de los AEs para el diseño y mejora de GPAIS, se alinean las propiedades de GPAIS con las áreas de ML en las que los AEs han contribuido en buena manera, y se muestran casos prácticos como hitos en el desarrollo de AEs con GPAIS. La Sección 4 hace énfasis en los retos para obtener los beneficios de usar AEs para GPAIS y también estrategias que permiten abordar GPAIS con AEs, junto con las áreas de investigación adecuadas dentro de los AEs para aplicar estas estrategias. Finalmente, en la Sección 5, se exponen las conclusiones extraídas y las líneas futuras de investigación.

1.1 Motivación

El ML es un subcampo de la IA que se centra en el desarrollo de modelos capaces de aprender patrones a partir de datos. La optimización de estos modelos ha sido un área destacada de investigación, donde se han obtenido resultados significativos [STÖ19] gracias a la adaptación de su diseño estructural y/o hiperparámetros en función de diversos objetivos, como el rendimiento, la complejidad o la robustez, entre otros [YS20, YOY+23]. La diversidad de objetivos de optimización considerados hasta la fecha en la optimización de modelos ML refleja la capacidad de estos algoritmos para abordar diferentes criterios.

Los avances recientes en varias áreas de investigación del ML como el Aprendizaje Profundo (*Deep Learning* - DL) [GBC16] y los Modelos de Lenguaje Grande (*Large*

Language Models - LLM) [MSC+13, HM15] – con chatbots que presentan un rendimiento sin precedentes como ChatGPT [VDBZ+23] y modelos entrenados para generar código que mejore la eficacia de los operadores de mutación dentro de la computación evolutiva [LGJ+24] – indican un cambio notable hacia sistemas de IA más generalizados. Dichos GPAIS han ganado importancia debido a sus capacidades únicas, no solamente para ejecutar las tareas de modelado para las que fueron diseñados originalmente, sino también para realizar bien aquellas tareas para las que no fueron entrenados explícitamente. Esta capacidad de generalización que va más allá de las tareas conocidas ha sido constatada por las recientes definiciones que se han atribuido a los GPAIS (véase [TMP+24] y las referencias allí comentadas).

Como se ha señalado anteriormente, existen numerosos enfoques para optimizar los modelos de ML que han demostrado una notable competitividad en diversos dominios del ML [SCZZ20]. Sin embargo, estudios recientes señalan una nueva tendencia de investigación centrada en la optimización de GPAIS, yendo más allá del ML tradicional [GAU+23, UGT23, CL23, BNN+23, TMP+24]. Áreas de interés dentro de la optimización como *open-ended evolution* [Sta03, TBC+16, PBC+19, ZLSC23], *quality-diversity optimization* [PSS16, CD18, BDT+23] y *novelty search* [LS08] se han convertido en una parte esencial para GPAIS. En dichas áreas, los GPAIS se sitúan a la vanguardia de la investigación actual. Estos sistemas presentan características atractivas, como la generación de nuevos conocimientos, la adaptación a diversos entornos, y la integración de nuevas tareas sin reducir el rendimiento. Al igual que ocurre con otros sistemas complejos, en los GPAIS hay decisiones críticas que afectan tanto a su diseño como a su optimización, y que abordan retos como la dimensionalidad del espacio de búsqueda, la evolución de los objetivos y la necesidad de adaptación continua. Este panorama dinámico ofrece oportunidades para integrar los GPAIS con otras técnicas de optimización eficaces.

En este sentido, los AEs destacan como una familia versátil de algoritmos que han liderado el diseño y optimización de modelos de ML [ASBC+19, TTBG21], teniendo un impacto sustancial en ambos escenarios [ÜB22, LMX+23]. Los AEs han desempeñado un papel fundamental en diversas áreas de investigación, aportando contribuciones valiosas al desarrollo de modelos de ML de alta calidad. Un ejemplo notable de su éxito está en su fusión con los LLM [AST+24].

Los AEs han demostrado su capacidad para evolucionar programas, resolver problemas de optimización dinámica o equilibrar varios objetivos en conflicto, entre otros escenarios. Además de su historial de éxitos en la optimización de sistemas de IA de propósito fijo, los AEs resultan especialmente atractivos para la optimización de GPAIS y para abordar los mayores retos que plantean en comparación con los sistemas de IA de propósito fijo. De hecho, ciertas áreas de investigación de los AEs se corresponden con algunas de las propiedades fundamentales de los GPAIS, como la adaptabilidad a problemas a lo largo del tiempo (optimización dinámica evolutiva) o la confluencia de múltiples objetivos en entornos multitarea (lo que corresponde a la optimización multitarea y multiobjetivo).

Este trabajo analiza el potencial de la IA que potencia a otra IA, utilizando a los AEs como una capa adicional de abstracción para diseñar o enriquecer GPAIS. Nos referiremos

a estos modelos como EA-GPAIS. En concreto, los objetivos de este trabajo son:

- *Relacionar el potencial de los AEs con los GPAIS*, con el fin de fomentar una mayor investigación en esta área de la IA. Para ello, desarrollaremos los tres objetivos siguientes:
  - Estudiar el diseño y la mejora de GPAIS mediante AEs con una taxonomía basada en AEs que potencia a otra IA. Cada categoría de la taxonomía sirve de guía sobre cómo los AEs pueden diseñar o mejorar la IA.
  - Asociar las propiedades de GPAIS con áreas específicas de ML en las que los AEs han realizado aportaciones significativas, para demostrar que la conexión entre los AEs y GPAIS está bien definida, ya que los AEs pueden facilitar estas propiedades.
  - Mostrar los hitos recientes de EA-GPAIS. Bajo la definición de GPAIS presentada en la siguiente sección, ilustramos diversos trabajos que han contribuido al inicio y progreso de este campo en los últimos años, mostrando así el potencial de EA-GPAIS.
- *Debatir los retos que plantea el aprovechamiento de las ventajas de EA-GPAIS y las estrategias para hacer frente a los avances*, centrándose en los retos futuros. Para ello, lo desglosamos en los dos siguientes subobjetivos:
  - Estudiar los retos existentes para obtener los beneficios de los AEs en el contexto de GPAIS. Tanto para el diseño como para la mejora de GPAIS el uso de los AEs es una tarea compleja, y debemos asegurarnos de explotar el beneficio de su sinergia.
  - Explorar las estrategias que pueden aplicarse con los AEs para diseñar y mejorar los GPAIS. Estas estrategias servirán de guía para los desarrollos actuales y futuros de GPAIS utilizando AEs.

2 Antecedentes y trabajo relacionado

Esta sección hace un breve resumen de la importancia de la computación evolutiva en ML con el transcurso del tiempo y en la nueva era de la generación de modelos en IA en la subsección 2.1. A continuación, se explican varias características de GPAIS en la subsección 2.2. Por último, se muestra la relevancia del paradigma de la IA que potencia a otra IA en relación con GPAIS en la subsección 2.3.

2.1 La Computación Evolutiva en el Aprendizaje Automático

La computación evolutiva ha proporcionado mecanismos inteligentes capaces de optimizar modelos en diversos entornos. Como señalan Nan Li et al. [LMX+23], que revisaron más de quinientas propuestas, estos algoritmos han sido ampliamente utilizados para optimizar las diferentes etapas dentro del diseño total de los modelos de ML, incluyendo el preprocesado, postprocesado y el propio modelado. Cabe destacar que dentro de todas estas etapas, la computación evolutiva ha alcanzado una importancia considerable en ramas específicas, como la selección de características [XZBY16], la extracción de características [MSNM21], el aprendizaje por conjuntos (*ensemble learning*) [Hey24], e incluso en la mejora del diseño de estos modelos mediante otras técnicas como las máquinas de vectores soporte (*Support Vector Machines*) [FLSL19] y los árboles de decisión (*Decision Trees*) [BBdCF12].

El rápido crecimiento del DL en los últimos años ha ampliado sus límites de aplicación. Se han desarrollado diversas propuestas evolutivas que han facilitado la optimización de los pesos, la arquitectura y la configuración de los modelos de DL [DSOM+19, ZLZ22]. La computación evolutiva ha contribuido significativamente al progreso en cada una de las nuevas temáticas del ML, gracias al desarrollo de algoritmos que impulsan el conocimiento en estas áreas. Estos algoritmos han sido ampliamente estudiados debido a su aplicabilidad en una gran variedad de disciplinas [GL23]. Por lo tanto, en la era moderna de la IA, en la que surgen problemas cada vez más complejos, los antecedentes de la computación evolutiva nos sugieren que está destinada a convertirse en uno de los mecanismos principales para la nueva generación de modelos de IA.

2.2 GPAIS: Definiciones y Propiedades

El estudio de los GPAIS parte de una clara distinción de los sistemas de IA de propósito fijo, que están diseñados para realizar tareas específicas. Esta diferenciación ha generado debates sobre las características que ha de presentar sistema de IA para ser clasificado como GPAIS [UGT23]. Un estudio reciente ofrece un análisis exhaustivo de los GPAIS, planteando una definición como referencia clara, una caracterización y una clasificación de estos sistemas mediante el uso de una taxonomía que distingue varias estrategias para construir GPAIS. En concreto, los GPAIS se definen en función del proceso seguido para

incorporar una nueva tarea al sistema, ya sea reentrenando todo el sistema desde cero con esa tarea o bien mediante una adaptación para solucionarla conservando el conocimiento aprendido por el sistema a partir de datos anteriores. Según [TMP+24]:

> *Definición 1 (GPAIS):* Un Sistema de Inteligencia Artificial de Propósito General (GPAIS) se refiere a un sistema avanzado de IA capaz de realizar eficazmente una serie de tareas distintas. Su grado de autonomía y habilidad viene determinado por varias características claves, como la capacidad de adaptarse o desenvolverse bien en nuevas tareas que surjan en el futuro, la demostración de ser competente en dominios para los que no ha sido entrenado de forma intencionada y específica, la capacidad de aprender a partir de datos limitados y el reconocimiento proactivo de sus propias limitaciones con el fin de mejorar su rendimiento.

La Definición 1 [TMP+24] proporciona una descripción exhaustiva de las características que puede poseer un GPAIS. A continuación, y basándonos en [TMP+24], presentamos brevemente los conceptos de GPAIS que nos ayudarán a entender el papel que pueden desempeñar los AEs.

El rasgo clave que distingue los GPAIS frente a la IA clásica de propósito fijo radica en la capacidad de abordar simultáneamente múltiples tareas de aprendizaje (ya sean conocidas o desconocidas). Teniendo en cuenta lo que sabemos sobre esas tareas, Triguero et al. [TMP+24] diferencian entre dos tipos de GPAIS:

- ***GPAIS de mundo cerrado***: Asumen que tenemos datos para un número determinado de tareas, y que esas serán las únicas tareas que se tratarán.
- ***GPAIS de mundo abierto***: Estos sistemas reconocen que pueden surgir nuevas tareas y que los datos disponibles pueden ser limitados (o inexistentes).

En el contexto del mundo abierto, se espera disponer de muy pocos datos, por lo que resulta fundamental aprovechar los conocimientos previos de otras tareas. De este modo, los GPAIS de mundo abierto se centran en la diversidad/generalización de modelos en vez de en el ajuste de la configuración del modelo para una tarea concreta, de modo que puedan utilizarse en diferentes problemas incluso en ausencia de suficientes datos para estas nuevas tareas. El conocido campo de la IA Generativa (GenAI) se compone de sistemas que son un ejemplo excelente de sistemas GPAIS de mundo abierto, que se caracterizan por su autonomía, adaptabilidad a nuevas tareas, ser competentes en dominios para los que no han sido entrenados expresamente, capacidad de aprender a partir de datos limitados y reconocer proactivamente sus propias limitaciones. Estas características han supuesto una transformación completa de la IA, como se refleja en [Mii24].

Esta definición nos permite distinguir entre varios grados de autonomía para los GPAIS, sin embargo, existen múltiples líneas de investigación para realizar estos sistemas. En la literatura especializada, encontramos dos enfoques distintos, pero no excluyentes.

Por un lado, se puede intentar crear un único modelo de aprendizaje multitarea que sea lo suficientemente general/diverso. Para ello, el modelo puede requerir ser entrenado sobre datos de carácter general (por ejemplo, mediante autosupervisión) para posteriormente ser adaptado ("fine-tuned") a nuevas tareas específicas. Esa es la idea subyacente de los modelos fundacionales (*foundation models*) [BHA+22]. Por otro lado, se puede optar por añadir una nueva capa de abstracción que utilice otra IA para ayudar a la IA subyacente a ser más general. Aquí es donde los AEs pueden marcar la diferencia. La siguiente subsección se centra en describir las diferentes vías de aplicación en las que un modelo de IA puede ayudar a otro basándose en la taxonomía propuesta en [TMP+24].

2.3 IA que potencia a otra IA para GPAIS

El concepto de modelos de IA potenciados o incluso diseñados por otro modelo de IA se conoce típicamente como IA que potencia a otra IA. Siguiendo la taxonomía propuesta en [TMP+24], mostramos una lista no exhaustiva de temas en los que una IA puede ser útil para diseñar o mejorar otra IA. En cuanto al diseño de un modelo general de IA, podemos optar por los siguientes niveles:

- **Optimización de hiperparámetros**: Aunque el uso de AEs para la optimización de hiperparámetros no es nuevo ni inusual en la IA de propósito fijo [YZ20, BKvS+23], aquí nos centramos en la idea de ajustar un conjunto de hiperparámetros para resolver una serie de tareas. En el contexto de mundo abierto, el reto consiste en encontrar la mejor configuración cuando surge una nueva tarea, potencialmente con pocos o ningún dato, para explotar el conocimiento previo con el fin de generalizar.
- **Selección automática de algoritmos**: El planteamiento anterior puede mejorarse aún más si se determina tanto el algoritmo de IA más adecuado como sus hiperparámetros. Esto permitiría una solución de IA más general que decide automáticamente la técnica de IA a utilizar para uno o varios problemas a la vez. En ML, esto se conoce normalmente como AutoML [HKV19]. En este campo, los AEs han hecho una buena aportación con el objetivo tanto de reducir la complejidad del modelo de DL como de obtener mejores resultados [MDVR+21, ZLZ22].
- **Construcción de algoritmos**: En un nivel más bajo, podemos utilizar un modelo de IA para diseñar los componentes de un algoritmo. AutoML-zero es un ejemplo notable de este tipo [RLSL20]. La neuroevolución [SCLM19] también es una alternativa para el diseño de modelos enteros de DL desde cero. Esto tiene el potencial de diseñar algoritmos que generalicen múltiples tareas directamente, pero a día de hoy se conoce muy poco trabajo para diseñar algoritmos desde cero que permitan realizar nuevas tareas.

El objetivo de enriquecer una IA con la entrada de otra IA suele ser para abordar problemas intrínsecos para que los modelos generalicen bien. Según [TMP+24], los siguientes cinco enfoques para enriquecer son clave en GPAIS.

- **Descubrir nuevos comportamientos** para hacer frente a los cambios dinámicos del entorno, como los cambios en la distribución de los datos subyacentes. El aprendizaje continuo (*continual learning*) [PKP+19] es un ejemplo destacado de este ámbito.

- **Generación de datos** para mitigar la falta de datos que puedan parecerse a los que podemos encontrar para una nueva tarea. Con la aparición de GenAI, los modelos generativos [SWVN23] pueden ser capaces de crear datos que sigan la distribución de un conjunto de datos de entrenamiento dado. POET [WLCS19] es un ejemplo relevante de generación de entornos/escenarios para mejorar la generalidad de un modelo en posibles escenarios de mundo abierto.

- **Aprendiendo a aprender** para compensar, de nuevo, la falta de datos para una tarea concreta en presencia de muchas tareas (a veces relacionadas). Aprender a transferir conocimiento de manera eficaz de un conjunto de tareas a una nueva tarea [WRH17] puede permitir que los GPAIS generalicen bien. *Few-shot learning* [WYKN20] es un ejemplo destacable en esta área.

- ***Active learning*** para buscar de forma autónoma la ayuda humana para perfeccionar un modelo existente [FYHT17]. Esto puede proporcionar una adaptabilidad más segura a nuevas tareas en situaciones desconocidas o inciertas.

- **Aprendizaje cooperativo y colectivo** puede utilizarse para construir sistemas más amplios compuestos por diferentes modelos de IA, que se enriquezcan mutuamente con distintas ideas. Un ejemplo es la explotación de diferentes modalidades de datos.

3 Relación entre el potencial de los AEs y los GPAIS: Taxonomía para AEs que potencia a otra IA y principales hitos

Como se ha comentado anteriormente, la implementación de GPAIS puede realizarse añadiendo una nueva capa de abstracción para diseñar o mejorar otra IA. Para mostrar el potencial de los AEs con los GPAIS, la subsección 3.1 y 3.2 describen cómo los AEs pueden ayudar a diseñar y mejorar los GPAIS, respectivamente. Además, la subsección 3.3 analiza la idoneidad de los AEs en GPAIS examinando sus propiedades y su conexión con áreas de investigación de ML en las que la computación evolutiva ha tenido un papel relevante. Finalmente, la subsección 3.4 muestra varios hitos de EA-GPAIS durante los últimos años.

3.1 AEs que potencia a otra IA para el diseño de GPAIS

Partiendo de la revisión realizada en la sección 2.3, exploramos ahora cada una de las categorías y examinamos las estrategias en las que los AEs han hecho sus aportaciones, sirviendo de referencia para futuros desarrollos GPAIS que utilicen AEs. A continuación, las describimos brevemente para destacar el potencial de los AEs en estos escenarios de diseño:

- **Optimización de hiperparámetros**: Esta categoría ha sido objeto de un amplio estudio a lo largo de los años, con los algoritmos genéticos apareciendo como un enfoque ampliamente utilizado para abordar el problema [DFDF$^+$18]. Sin embargo, existen heurísticas alternativas como la Optimización por Enjambre de Partículas (*Particle Swarm Optimization*) o la Optimización Bayesiana, que también demuestran su eficacia en este contexto [YS20]. En el ámbito del DL, el DL evolutivo se centra normalmente en descubrir el conjunto óptimo de hiperparámetros [DSOM$^+$19, DHD20, BBS20]. Los métodos de*Neural Architecture Search* (NAS) evolutiva son particularmente adecuados para identificar las mejores configuraciones de hiperparámetros para estos modelos [OSB$^+$14]. Además, los últimos avances en neuroevolución han integrado NAS y AEs co-evolutivos, incorporando los hiperparámetros como parte de este proceso evolutivo [MLM$^+$24a].

- **Selección automática de algoritmos**: El área de NAS evolutiva en DL ha recibido una atención considerable en los últimos años, con varias versiones de Algoritmos Genéticos que se han utilizado para seleccionar la estructura óptima para los modelos de DL [SXZ$^+$20, SXZY20]. Su capacidad para explorar grandes espacios de búsqueda los hace muy adecuados para estas tareas, como se destaca en [LSX$^+$23, ZLZ22] tanto para los AEs monoobjetivo como multiobjectivo. Es importante señalar que NAS a menudo abarca tanto la selección de modelos como

la optimización de hiperparámetros, y muchas propuestas las tratan como si fueran la misma tarea.

- **Construcción de algoritmos**: En esta categoría, las propuestas de NAS evolutiva son capaces de diseñar modelos de DL en una amplia gama de dominios [EMH19b, ZQGT21]. Además, la Programación Genética se ha utilizado no solamente para los procesos de selección y construcción de características [EVH10], sino también para la construcción de características por sí sola [TXZ16]. Por último, AutoML-zero proporciona un método evolutivo para construir redes neuronales desde cero [RLSL20].

3.2 AEs que potencia a otra IA para la mejora de GPAIS

Esta sección se centra en cómo pueden aprovecharse los AEs para mejorar los GPAIS. La necesidad de capacidades de generalización y adaptabilidad de los GPAIS son dos cuestiones clave para las que los AEs han mostrado ser exitosos. El enriquecimiento de los GPAIS mediante el uso de los AEs puede llevarse a cabo de la siguiente manera:

- **Descubrir nuevos comportamientos**: Para que los GPAIS se adapten eficazmente a entornos dinámicos, la optimización dinámica evolutiva se perfila como un área esencial, ofreciendo propuestas innovadoras [NYB12]. Además, la adaptabilidad es mayor cuando se combina con AEs multiobjetivo [ABBS17].

- **Generación de datos**: En situaciones en las que los GPAIS se enfrentan a datos limitados, el objetivo principal de los AEs es extraer toda la información de calidad del conjunto de datos disponibles. La generación evolutiva de datos se emplea ampliamente en diversos ámbitos, incluso para problemas con desbalanceo de datos [KGJH16]. Además, *open-ended evolution* no solamente genera datos, sino que también crea entornos de aprendizaje, enriqueciendo la comprensión del modelo [WLCS19]. Los AEs también pueden generar diversas configuraciones del modelo inicial para adaptar el modelo a los datos [CCH$^+$23]. Por último, *quality-diversity optimization* ofrece un enfoque global para esta categoría. Estos métodos garantizan la generación de una colección diversa de individuos con el mayor rendimiento posible, presentando también propiedades deseables como la robustez y la adaptabilidad a diversos escenarios [PSS16].

- **Aprendiendo a aprender**: Dado que los GPAIS suelen funcionar con datos limitados, resulta imprescindible aprovechar la información de tareas similares. El aprendizaje por transferencia evolutivo (ETL) y la optimización por transferencia evolutiva (ETO) no solamente permiten transmitir los parámetros aprendidos, sino también las representaciones y los operadores [TFJ21]. Además, la integración de la optimización dinámica evolutiva con el aprendizaje de transferencia expande el espectro de aplicaciones en escenarios de GPAIS [JWQ$^+$21, WWZ$^+$23].

- ***Active learning***: Este enfoque se ha analizado junto con la optimización multiobjetivo, dando lugar a métodos capaces de aprovechar el conocimiento y descubrir soluciones de alta calidad [RV18]. El frente de Pareto delimita una región dentro del espacio de búsqueda donde se encuentran las buenas soluciones, enriqueciendo así el modelo mediante la localización de soluciones de alta calidad [LJJ22]. Además, la integración del *active learning* con otros enfoques basados en AEs, como los AEs co-evolutivos, puede reforzar aún más las interacciones y conducir a mejores soluciones [LLL14].

- **Aprendizaje cooperativo y colectivo**: La optimización multitarea evolutiva (EMO) [ODSMH22] facilita la maximización del conocimiento entre tareas, permitiéndoles aprender unas de otras [XQX22]. Los AEs desempeñan un papel crucial en la mejora de este proceso, particularmente en escenarios de GPAIS donde las tareas pueden variar en cuanto a su tipo, permitiendo al AE explotar la sinergia de las interacciones. Además, los AEs co-evolutivos contribuyen a la explotación de la información facilitando su intercambio entre poblaciones [MLZ+19]. El *ensemble learning* aprovecha el aprendizaje colectivo de modelos individuales para explotar la sinergia entre tareas [Hey24]. Este enfoque permite al sistema aprovechar el conocimiento grupal para mejorar el rendimiento. Cabe señalar que esta categoría no sólo mejora las prestaciones existentes, sino que también tiene el potencial de descubrir nuevos comportamientos a través de la cooperación y el intercambio de información, mejorando aún más el sistema.

3.3 Conexión entre GPAIS, Aprendizaje Automático, y Áreas de Investigación sobre Optimización

En esta sección, analizamos las capacidades de los AEs para los GPAIS teniendo en cuenta sus diversas necesidades. Examinamos qué propiedades de los GPAIS pueden ser compatibles con los resultados y los conocimientos obtenidos en las distintas áreas de investigación de los AEs. Complementamos este análisis con una evaluación de las capacidades de las áreas de investigación en AEs para los GPAIS considerando sus propiedades:

- **GPAIS han de adaptar sus conocimientos a tareas que varían con el tiempo aprovechando los conocimientos previos**: Para estos escenarios, los AEs para optimización dinámica podrían ser muy útiles, ya que es necesario ajustar la estrategia de búsqueda del algoritmo en tiempo de ejecución para poder adaptar la búsqueda al panorama cambiante de la optimización. En concreto, se podrían detectar nuevos patrones a lo largo del tiempo, o decidir ignorar patrones anteriores que ya no reflejen las tareas actuales.

- **GPAIS pueden realizar varias tareas simultáneamente, incluso siguiendo diferentes prioridades u objetivos**: Los AEs multiobjetivo podrían utilizarse

para optimizar los GPAIS teniendo en cuenta distintos objetivos y obteniendo modelos con distintos equilibrios entre ellos. Asimismo, los AEs multitarea pueden aprender diferentes tareas simultáneamente, lo que permite mejorar los GPAIS al abordar múltiples tareas. Otra área de los AEs que podría aplicarse es la de los algoritmos co-evolutivos, mediante los cuales muchos algoritmos se ejecutan en paralelo e intercambian información para mejorar la búsqueda. Este enfoque co-evolutivo podría ayudar a los GPAIS para explotar las sinergias entre las tareas compartiendo conocimientos entre ellas.

- **GPAIS pueden realizar tareas inéditas con pocos o ningún dato nuevo**: Para reforzar esta funcionalidad, los AEs se pueden emplear para optimizar simultáneamente múltiples modelos de IA, garantizando la diversidad entre sus conocimientos modelados. Esta diversidad abarca multitud de opciones, lo que aumenta la probabilidad de identificar modelos que muestren un rendimiento superior en tareas nuevas y desconocidas.

- **GPAIS deben poder construirse/configurarse de forma autónoma**: Los AEs se han empleado tradicionalmente para el ajuste automático de modelos de ML en tres niveles de granularidad diferentes. En el nivel más alto, los AEs se utilizan para la selección de algoritmos, un proceso en el que se han aplicado ampliamente. Pasando a un nivel intermedio, una vez determinado el algoritmo, los AEs pueden configurar sus parámetros para optimizar el rendimiento. Por último, en el nivel más bajo de granularidad, los AEs participan en la configuración o construcción de primitivas de modelos, lo que demuestra su versatilidad en todos los niveles de ajuste de modelos.

- **GPAIS debe funcionar eficazmente en el diseño, en la fase de entrenamiento y en el proceso de inferencia**: El problema del rendimiento ha sido motivo de preocupación para los AEs y se ha estudiado con diferentes técnicas que pueden trasladarse a otros contextos. Por ejemplo, para mitigar los costes de entrenamiento, se puede emplear un AE para reducir el conjunto de datos de entrenamiento, creando un conjunto de datos reducido con un rendimiento similar, una técnica conocida como destilación de datos. Además, los AEs pueden utilizarse para reducir la complejidad del modelo, ya sea mediante la poda o la cuantización, disminuyendo así los costes de entrenamiento y de inferencia.

- **GPAIS deben explorar, evaluar y decidir acciones o secuencias de acciones en pos de metas u objetivos específicos**: Este es un escenario común para los AEs. La computación memética, es una alternativa sólida y robusta para la búsqueda en dominios complejos gracias a su enfoque combinado de exploración-explotación. Además, los AEs se han utilizado tradicionalmente para el aprendizaje por refuerzo, en el que los resultados de las acciones se utilizan para reforzar las mejores acciones, una técnica que se aplica en diversos ámbitos, desde los juegos hasta la robótica. Por consiguiente, los AEs pueden ofrecer robustez y adaptabilidad incluso en entornos dinámicos.

- **GPAIS debe abordar simultáneamente tareas de modelización multimodal definidas sobre diversos conjuntos de datos**: Los AEs demuestran su versatilidad a la hora de manejar sin problemas diversas representaciones. Los AEs adaptados a determinadas representaciones pueden trabajar en cooperación para abordar eficazmente tareas multimodales. En consecuencia, la multimodalidad del conocimiento puede mejorarse mediante los AEs, facilitando la integración de información heterogénea en un espacio de características unificado.

- **GPAIS deben aprender de forma cooperativa, intercambiando conocimientos sobre las tareas aprendidas y aprovechando las sinergias que se deriven de ellas**: Se han desarrollado numerosas técnicas para facilitar el intercambio de información entre modelos, como los AEs co-evolutivos o la optimización multitarea.

- **GPAIS son capaces de estimar su confianza a la hora de abordar su(s) tarea(s) y de solicitar proactivamente nueva información cuando no están seguros de su resultado.**: En los casos en que se carezca de información suficiente, los GPAIS deberían decidir de forma autónoma solicitar datos adicionales. Sin embargo, para minimizar la demanda de nuevos datos y garantizar su eficacia, estos deben seleccionarse con criterio y distribuirse ampliamente. Los AEs pueden generar prototipos de nuevos datos y seleccionar entre ellos en función de su contribución potencial al conjunto de datos existente. Además, un experto puede supervisar y evaluar las distintas opciones propuestas por el AE. Este enfoque permite al experto centrarse en la evaluación de las opciones propuestas, en lugar de limitarse a proponer la necesidad de nueva información.

Las Tablas 1 y 2 resumen los resultados de este análisis, emparejando cada propiedad de un GPAIS con su área de investigación de optimización de ML relacionada, y facilitando un listado no exhaustivo de las áreas de estudio de computación evolutiva que se conectan con cada propiedad. En conclusión, los EA-GPAIS se convierten en la extensión lógica del ML evolutivo tradicional, donde los AEs se han aplicado tradicionalmente para diseñar y optimizar el rendimiento de los sistemas de ML.

Table 1: Conexión entre las propiedades de GPAIS, el ML y áreas de investigación de la computación evolutiva, junto con la motivación para EA-GPAIS.

Propiedad GPAIS	**Área del ML**	**Área sobre AEs**	**Motivación para EA-GPAIS**
GPAIS adaptan sus conocimientos a tareas que varían con el tiempo aprovechando los conocimientos previos	*Continual learning*, *data shift*, *concept drift*, aprendizaje por transferencia	Optimización dinámica evolutiva	• Aparición/desaparación de nuevos patrones • Cambios graduales/drásticos de tareas con el tiempo
GPAIS pueden realizar varias tareas simultáneamente, explotando sinergias entre ellas	Aprendizaje multitarea, aprendizaje por transferencia, meta-aprendizaje	Optimización multiobjetivo, optimización evolutiva multitarea, co-evolución cooperativa	• Conflictos en la naturaleza de diferentes objetivos $\mapsto$ necesidad de balance entre ellos • Intercambio de conocimiento entre GPAIS para abordar diferentes tareas.
GPAIS pueden realizar tareas inéditas con pocos o ningún dato nuevo	*Zero-/Few-shot learning*	*Open-ended evolution, quality-diversity optimization*, AEs multimodales, generación evolutiva de datos	• Modelado de lo desconocido en GPAIS de mundo abierto $\mapsto$ Diversificación del conocimiento del modelo
GPAIS deben poder construirse/configurarse de forma autónoma	AutoML, selección/diseño automático de algoritmos, meta-aprendizaje	NAS evolutivo, programación genética, hiper-heurísticas	• Configuración de las primitivas de bajo nivel • Espacios de búsquedas más grandes
GPAIS debe funcionar eficazmente en el diseño, en la fase de entrenamiento y en el proceso de inferencia	Métodos de validación de modelos de ML, compresión de modelos (poda/cuantización), destiliación del conocimiento y de los datos, dispersión estructurada	Poda/cuantización evolutiva, optimización global de gran escala, AEs paralelos, AEs distribuidos, optimización basada en sustitutos, asignación dinámica de recursos en los AEs	• Evaluación del modelo $\mapsto$ Alto coste computacional • Variabilidad de tareas respecto al tiempo puede requerir una reconfiguración a pesar del tiempo extra de entrenamiento

Table 2: Conexión entre las propiedades de GPAIS, el ML y áreas de investigación de la computación evolutiva, junto con la motivación para EA-GPAIS (II).

Propiedad GPAIS	**Área del ML**	**Área sobre AEs**	**Motivación para EA-GPAIS**
GPAIS deben explorar, evaluar y decidir acciones o secuencias de acciones en pos de metas u objetivos específicos	Árbol de búsqueda Monte Carlo, aprendizaje por refuerzo (multiagente), programación dinámica, A*	Aprendizaje por refuerzo evolutivo, teoría de juegos evolutiva, robótica evolutiva	• Compensación de exploración-explotación para descubrir estrategias óptimas (exploración) mientras que se explotan estrategias conocidas para maximizar la recompensa (explotación) • Muestreo eficaz del espacio de estrategias posibles • Robustez y adaptabilidad en configuraciones dinámicas
GPAIS debe abordar simultáneamente tareas de modelización multimodal definidas sobre diversos conjuntos de datos	Fusión de datos, aprendizaje multimodal, aprendizaje de representaciones, recuperación multimodal, adaptación del dominio	AEs mixtos, aprendizaje de representaciones evolutivo, AEs co-evolutivos, programación genética, conjuntos de AEs	• Integración y codificación de información heterogénea de diferentes modalidades dentro de un espacio de características • Adaptabilidad robusta a las características variables de los datos
GPAIS deben aprender de forma cooperativa, intercambiando conocimientos sobre las tareas aprendidas y aprovechando las sinergias que se deriven de ellas	Adaptación del dominio, aprendizaje por transferencia, aprendizaje federado	Optimización multitarea evolutiva, AEs distribuidos, AEs co-evolutivos	• Estrategias sencillas de intercambio de conocimientos entre AE se pueden aplicar a las tareas relacionadas con el modelado GPAIS
GPAIS son capaces de estimar su confianza a la hora de abordar su(s) tarea(s) y de solicitar proactivamente nueva información cuando no están seguros de su resultado.	Estimación de la incertidumbre, modelado bayesiano, *active learning*	Aumento de datos evolutivo, generación de prototipos evolutiva, selección de prototipos evolutiva	• AEs pueden utilizarse para optimizar la selección de puntos de datos cuya supervisión se consulta al oráculo • AEs pueden utilizarse para seleccionar un subconjunto diverso y representativo de datos no etiquetados para su etiquetado

3.4 Hitos y logros notables de EA-GPAIS

En esta sección, analizamos con más detalle casos específicos representados por propuestas que han contribuido significativamente al avance del campo en los últimos años. Ilustramos diferentes trabajos registrados en la literatura para EA-GPAIS de mundo cerrado y abierto en las subsecciones 3.4.1 y 3.4.2, respectivamente. La Tabla 3 muestra estas propuestas exitosas en diferentes áreas. Las dos primeras filas corresponden a propuestas de GPAIS de mundo cerrado en las que no existen mecanismos de adaptación a nuevas tareas. Aunque estas propuestas representan un avance hacia los GPAIS de mundo abierto, siguen estando en un escenario cerrado. La última fila, separada con una línea más gruesa, está compuesta por enfoques evolutivos en GPAIS de mundo abierto, ya que incorporan mecanismos para generar diversidad y compartir conocimientos.

3.4.1 Hitos recientes en los EA-GPAIS de mundo cerrado

Las propuestas enumeradas en esta tabla han marcado un rumbo significativo en la investigación, sentando las bases iniciales en la sinergia entre AEs y GPAIS. Muchos trabajos sobre NAS, DL evolutivo y neuroevolución han impulsado avances importantes en estos ámbitos, como demuestran los estudios exhaustivos publicados a lo largo de los años [TSK+18, ASBC+19, TTBG21, ÜB22, LMX+23, LSX+23, MDVR+21, DHD20, ZLZ22]. Los puntos comunes entre estas propuestas son el enfoque en la configuración del modelo, mediante el uso de un AE para evolucionar pesos o hiperparámetros de un GPAIS basado en redes neuronales. Comenzamos describiendo varios casos de estudio relacionados con la optimización de hiperparámetros:

- **NEAT** [SM02] es la primera etapa en el campo de la neuroevolución. Utiliza un AE para hacer evolucionar redes neuronales mínimas hacia la creación de arquitecturas de red más profundas.

- **EDEN** [DB17] propone evolucionar diferentes tipos de capas convolucionales, de *pooling* y *fully-connected* con sus hiperparámetros dentro de las redes neuronales profundas.

- **EvoAAA** [CRMdJ20] es una propuesta de NAS vinculada a los *autoencoders*. En este caso, la configuración del modelo (arquitectura, pesos e hiperparámetros) evoluciona hacia una red con un mejor rendimiento en cuanto a la precisión.

- **DENSER** [ALMR19] pertenece a una rama de propuestas en NAS que se caracterizan por el uso de una gramática de operadores para evolucionar redes neuronales. DENSER utiliza un enfoque genético que codifica la macroestructura de la red neuronal (capas, tasa de aprendizaje, parámetros, etc.), mientras que la evolución gramatical especifica los parámetros de cada unidad del algoritmo evolutivo y el rango válido de los parámetros.

Table 3: Enfoques evolutivos para GPAIS de mundo cerrado y de mundo abierto.

GPAIS	Objective	Hyper-parameter optimization	Algorithm selection	New algorithm construction
Closed-world	Performance	NEAT EDEN EvoAAA DENSER	LSEIC CoDeepNEAT LEAF A-MFEA-RL	AutoML-zero
	Performance & Complexity	NSGA-Net NSGANetv2 NAT	LEMONADE MOENAS-TF-PSI	MOAZ
Open-world	Diversity	POET EGANS EUREKA XferNAS ESBMAL	–	–

La categoría de selección de algoritmos difiere de la categoría anterior. En esta segunda categoría se hace hincapié en la búsqueda del mejor algoritmo más que en la mejor configuración de hiperparámetros. En NAS y DL evolutiva, los autores a menudo solapan estas categorías, ya que la arquitectura de la red suele codificarse como parte de los hiperparámetros que deben optimizarse. Por lo tanto, la evolución de la arquitectura de red puede producir el mejor *algoritmo* (red neuronal) para abordar el problema en cuestión. Destacamos varios trabajos bajo esta categoría que han alcanzado una gran influencia en esta rama de la literatura:

- **CoDeepNEAT** [MLM+24a] representa uno de los avances más reputados en el campo del NAS. Se aplica un esquema coevolutivo con dos poblaciones de esquemas (la columna vertebral de la red neuronal) y los módulos que se insertarán en cada parte del esquema. A continuación, se ejecuta una búsqueda evolutiva para conseguir la mejor columna vertebral y los mejores módulos, junto con sus parámetros.

- **LSEIC** [RMS+17] proporciona una visión sobre la evolución de clasificadores a gran escala, con la intención de buscar automáticamente la mejor arquitectura para abordar el problema en cuestión. Para ello, este trabajo propone recurrir a varias

estrategias de codificación y operadores de mutación en el algoritmo evolutivo que sea aplica

- **LEAF** [LMH+19] es un entorno de trabajo basado en que usa internamente CoDeepNeat como motor interno para evolucionar redes. Constituye un marco en AutoML evolutivo que optimiza no solo los hiperparámetros, sino también las arquitecturas de red y su tamaño.

- **A-MFEA-RL** [MDSOH21] aprovecha la capacidad de la optimización evolutiva multitarea para configurar un GPAIS capaz de resolver simultáneamente múltiples escenarios de aprendizaje por refuerzo. Para ello, se realiza una búsqueda evolutiva sobre un espacio de búsqueda unificado que representa la arquitectura de la red neuronal, el número de neuronas de cada capa y la presencia de capas compartidas entre modelos.

Otras propuestas de NAS han optimizado métricas más allá de la precisión, centrándose especialmente en la complejidad de las redes. Estas propuestas comparten características similares con las anteriores, pero en su mayoría incorporan algoritmos evolutivos multiobjetivo para resolver dichos objetivos. A continuación resumimos algunos de los enfoques representativos más conocidos:

- **NSGA-Net** [LWB+19] es una aplicación de un algoritmo evolutivo multiobjetivo (NSGA-II) para encontrar las redes neuronales con el mejor equilibrio entre rendimiento de modelado y complejidad. NSGA-Net implica una exploración del espacio de arquitecturas potenciales de redes neuronales en tres pasos: un primer paso de inicialización de la población basado en el conocimiento previo de arquitecturas construidas a mano, un paso de exploración usando cruces y mutaciones en las arquitecturas, y un paso de explotación basado en un historial de arquitecturas neuronales evaluadas.

- **NSGANetv2** [LDG+20] amplía la propuesta anterior de NAS basándose en un algoritmo evolutivo multiobjetivo que utiliza dos sustitutos, uno a nivel de arquitectura y otro a nivel de pesos. En el nivel de arquitectura, el sustituto mejora la eficiencia de la muestra y en el nivel de pesos, los pesos evolucionan a través de una *Supernet* basada en las arquitecturas candidatas.

- **NAT** [LSG+21] presenta un mecanismo para el diseño automáticamente de redes neuronales, aprovechando el aprendizaje de transferencia con múltiples objetivos. Para lograr este objetivo, NAT utiliza *Supernets* de tareas específicas que comparten sus conocimientos con otras subredes dentro de un proceso de búsqueda evolutiva multiobjetivo. Aunque este planteamiento avanza hacia un GPAIS de mundo abierto al incorporar el intercambio de conocimientos, se trata de un GPAIS de mundo cerrado, ya que carece de mecanismos de adaptación a nuevas tareas.

Las dos propuestas siguientes se refieren a la Selección del algoritmo con varios objetivos, centrándose en la obtención de la propia red más que en la optimización de sus pesos:

- **LEMONADE** [EMH19a] utiliza un algoritmo evolutivo multiobjetivo para buscar arquitecturas bajo múltiples objetivos. La novedad radica en cómo LEMONADE aborda el consumo de recursos, utilizando un mecanismo de herencia lamarckiana. Este mecanismo genera redes hijas que parten del rendimiento predictivo de sus padres entrenados. Para ello, se aplican operadores de morfismos, utilizando un concepto similar al empleado en las propuestas anteriormente explicadas.
- **MOENAS-TF-PSI** [PL23] es otro AE multiobjetivo que pretende mejorar ciertas soluciones sobre frentes aproximados utilizando una búsqueda local denominada *potential solution improving*. Además, recurre a una métrica basada en la precisión como métrica libre de entrenamiento para estimar el rendimiento de la red evolucionada sin ejecutar ninguna época de entrenamiento, reduciendo el considerable coste computacional típico de los métodos NAS.

Este breve repaso a la literatura reciente sobre EA-GPAIS de mundo cerrado termina con la revisión de los esfuerzos en lo que se refiere a la construcción de nuevos algoritmos. En estos casos, el objetivo es crear un modelo completamente nuevo y su algoritmo de aprendizaje a partir de primitivas de procesamiento de bajo nivel. Dos propuestas recientes se encuadran en esta categoría:

- **AutoML-zero** [RLSL20] emplea un AE que altera el paradigma tradicional de los GPAIS de mundo cerrado al ir más allá de la optimización del rendimiento. No solamente descubre los hiperparámetros óptimos, sino que también desarrolla un algoritmo completo para crear un modelo adaptado a un problema de modelado dado.
- **MOAZ** [GAK+23] representa la variante multi-objetivo de AutoML-zero. El objetivo es distribuir las soluciones en un frente de Pareto compensando la precisión con la complejidad computacional del algoritmo. Además de generar diferentes soluciones Pareto-óptimas, MOAZ puede recorrer eficazmente el espacio de búsqueda para mejorar la eficiencia de la búsqueda utilizando operadores especializados de cruce y mutación.

Estos EA-GPAIS se han centrado predominantemente en la optimización del rendimiento de los GPAIS y en la adición de objetivos de búsqueda adicionales, como la complejidad. Sin embargo, aún queda camino por recorrer para que estos EA-GPAIS cumplan las propiedades que se les presuponen, especialmente en lo que se refiere a su naturaleza multimodal y multitarea y a su eficiencia computacional. Los logros de los GPAIS de mundo cerrado descritos hasta la fecha solamente consideran una única tarea, requieren elevados costes computacionales y suponen que se dispone de suficientes datos para la formulación de una función objetivo que evite que el GPAIS se ajuste en exceso.

3.4.2 Hitos recientes en los EA-GPAIS de mundo abierto

Las propuestas más recientes se han centrado en garantizar que los GPAIS puedan integrar nuevas tareas dentro de sus conocimientos adquiridos, a diferencia de los métodos tradicionales de búsqueda por objetivos que limitan la capacidad del sistema para integrar una nueva tarea. Aunque no existan estudios específicos sobre estos conceptos basados en la diversidad, varias áreas de investigación están estrechamente relacionadas con los principios de los GPAIS de mundo abierto. Los últimos estudios revisados, en lo que sigue, se basan precisamente en la generación de diversidad y el descubrimiento de comportamientos de modelos diversos:

- **POET** [WLCS19] se centra en la generación de diversidad a través de la síntesis de datos. En este trabajo, los agentes se emparejan con estos nuevos entornos generados para aprender de ellos y, al mismo tiempo, también se optimizan los pesos de los agentes. Además, pueden incluso transferirse de un entorno a otro, utilizando sus conocimientos para adaptarse al otro. La diversidad se induce a través de la síntesis de datos, donde el conocimiento se transfiere entre entornos a través de los agentes, optimizando en última instancia el modelo a través de los pesos de los agentes.
- **EGANs** [CCH$^+$23], enmarcado en el área de las Redes Generativas Adversarias, sirve como ejemplo de generación de diversidad a través del modelo, particularmente en un entorno de *zero-shot learning*. Mediante un enfoque evolutivo, EGANS aprende inicialmente el generador óptimo de modelos. En una etapa posterior, este generador pasa a formar parte de otro proceso evolutivo para determinar el modelo final.
- **EUREKA** [MLW$^+$23] constituye otro GPAIS de mundo abierto en el que se realizan varias tareas de aprendizaje por refuerzo al mismo tiempo. El algoritmo evolutivo hace evolucionar varias funciones de recompensa en un contexto basado en el código fuente del entorno. EUREKA genera funciones de recompensa ejecutables, mejorándolas con una búsqueda evolutiva que produce iterativamente lotes de recompensas candidatas.
- **XferNAS** [Wis20] presenta un marco de GPAIS de mundo abierto para la transferencia de conocimientos. XferNAS recoge el conocimiento de origen de múltiples tareas y combina este conocimiento para generar una arquitectura para una nueva tarea.
- **ESBMAL** [RV18] integra *Active Learning* y AEs para mejorar el etiquetado de datos. El AE reporta de forma óptima los mejores lotes de datos para potenciar el proceso de selección de instancias, contribuyendo al conjunto de propuestas de mundo abierto junto con los enfoques anteriormente mencionados.

Durante nuestra investigación, no hemos encontrado ningún trabajo específico centrado en la utilización de AEs para la selección o construcción de algoritmos en GPAIS de

mundo abierto. A diferencia de los GPAIS de mundo cerrado, el uso de nuevas métricas de diversidad para desarrollar GPAIS de mundo abierto puede considerarse un nicho de investigación aún por explorar.

Otra característica importante es que los GPAIS actuales suelen considerar una sola tarea en un entorno de mundo cerrado, lo que es aún más raro en los problemas de mundo abierto. Sorprendentemente, la aplicación de AEs se ha utilizado para el diseño y optimización de modelos de aprendizaje multitarea [ZNL+23]. También hemos destacado EUREKA como una propuesta de GPAIS de mundo abierto que trabaja con agentes de aprendizaje por refuerzo capaces de realizar varias tareas. Estos nichos de investigación que hemos detectado deberían estimular los esfuerzos en estudios venideros relacionados con GPAIS y AEs.

Los sistemas GenAI suelen ser más autónomos y no dependen de expertos. Aun así, el uso de AEs puede permitirles alcanzar niveles de autonomía aún mayores, reduciendo incluso de este modo la necesidad de contar con un experto. Utilizar otra IA para ayudar a mejorarla puede ser un enfoque viable. Dentro del campo del GenAI, cada vez son más los trabajos que utilizan AEs para potenciar este tipo de sistemas, lo que constituye uno de los principales focos de atención en los próximos años [WWW+24]. En esta línea de investigación, los trabajos recientes aprovechan los AEs como algoritmos clave para tareas como la fusión de modelos y la creación de modelos de base [AST+24], la evolución del código generado por los LLM [HMO24], la evolución de las salidas de los LLM [GWG+24], y la generación de algoritmos de optimización [LTYZ23].

4 Retos para aprovechar las ventajas de EA-GPAIS y estrategias para hacer frente a los progresos realizados

La integración de los AEs con los GPAIS plantea retos importantes, a pesar de sus beneficios potenciales. Una de las principales dificultades es garantizar la sinergia entre la naturaleza adaptativa de los AEs y los complejos procesos de toma de decisiones de los GPAIS. Por este motivo, en la subsección 4.1 recogemos estos retos de cómo los AEs son capaces de adaptarse a los GPAIS para el diseño y optimización de estos sistemas. Para superar estos desafíos, también presentamos varias estrategias que pueden implementarse con AEs ya sea para el diseño como para la mejora de los GPAIS. En concreto, nos centramos en su objetivo, su importancia en el contexto de GPAIS, y un análisis de las áreas de investigación basadas en AEs que pueden ayudar a materializar estas estrategias. Este análisis, respaldado a través de las secciones 4.2 a 4.5, sirve como muestra para motivar el desarrollo presente y futuro de EA-GPAIS.

4.1 Challenges harnessing the benefits of EA-GPAIS

Hay muchas formas en las que los AEs pueden beneficiar tanto a los GPAIS de mundo cerrado como a los de mundo abierto. Los AEs pueden mejorar los GPAIS de mundo cerrado facilitando el preprocesamiento avanzado de datos, optimizando los hiperparámetros y adaptando los modelos. Además, EMO utiliza los AEs para aprender simultáneamente varias tareas, lo que demuestra su versatilidad a la hora de abordar diversos retos de forma individual.

En escenarios de mundo abierto, los modelos deben adaptarse a nuevos problemas con un número mínimo de datos, garantizando un rendimiento sólido en los diversos retos que puedan surgir. Los AEs ofrecen funciones objetivo flexibles que refuerzan la solidez dentro del espacio de soluciones. La optimización multifactorial favorece la solidez entre tareas, mejorando potencialmente el rendimiento en nuevos problemas.

Los AEs ofrecen la ventaja de generar múltiples soluciones a lo largo del proceso de optimización, lo que garantiza una gama de modelos de IA optimizados para resolver nuevos problemas. El uso de una variedad de modelos de IA puede mejorar los resultados, ya que diferentes modelos pueden funcionar mejor para retos específicos. Mantener esta diversidad es importante para lograr resultados competitivos sin necesidad de un reentrenamiento exhaustivo.

4.2 EA-GPAIS mediante la explotación y adaptación de del conocimiento existente

4.2.1 ¿Por qué es importante esta estrategia para GPAIS?

El desarrollo de un sistema capaz de realizar un conjunto diverso de tareas requiere un esfuerzo considerable. Por lo tanto, todo el conocimiento que posea el sistema debe considerarse fundamental para su mejora continua. Los GPAIS deben aprovechar todos su conocimiento del conjunto de tareas existentes para mejorar el rendimiento y adaptarse a las nuevas tareas entrantes. Esto obliga al sistema a mantener su nivel de rendimiento en las tareas anteriores, al tiempo que se adapta con éxito a las nuevas que van apareciendo con el tiempo.

4.2.2 ¿Cómo pueden contribuir los AEs a esta estrategia?

Como ya se ha dicho previamente, ETO [TFJ21], EMO [ODSMH22], y ETL [ZQD$^+$21] han demostrado ser estrategias para optimizar el conocimiento adquirido previamente. Por lo tanto, mediante la generalización del aprendizaje a través de problemas, ETO, EMO y ETL pueden ser útiles para optimizar el intercambio de conocimientos entre los modelos concebidos para diferentes tareas.

4.2.3 ¿Qué áreas de investigación de los AEs son útiles para esta estrategia?

Los ámbitos de investigación de los AEs mencionados anteriormente pueden vincularse a la estrategia de explotación y adaptación de los conocimientos existentes:

- La ETO combina principios de los AEs y el aprendizaje por transferencia como esencia, el conocimiento o las soluciones aprendidas de un problema de optimización en un dominio de origen sirven para mejorar un proceso de optimización en otro dominio relacionado. Cuando las soluciones a tales problemas representan el conocimiento del GPAIS (es decir, sus parámetros aprendidos), ETO puede optimizar la adaptación del conocimiento transferido al problema de destino afinándolo o reconfigurándolo para que se adapte mejor a las características del dominio de destino.

- El ETL pretende aprovechar el conocimiento de dominios relacionados para mejorar el rendimiento en un dominio diferente, pero principalmente se centra en mejorar el aprendizaje y la adaptación del modelo mediante AEs. El ETL se ha utilizado ampliamente en NAS, con trabajos como XferNAS, en el que se aplica un proceso de aprendizaje por transferencia para iniciar arquitecturas para una nueva tarea, de forma que el conocimiento en otros problemas relacionados se transfiere para crear una red [Wis20]. Otro trabajo que ilustra el potencial de ETL se presenta en [PMMS$^+$23], donde el proceso de transferencia de conocimiento entre tareas en

dominios similares se acelera mediante la poda de los componentes innecesarios de la red neuronal utilizando un AE, por lo que el GPAIS puede generalizar de forma autónoma a otras tareas al tiempo que conserva el conocimiento de las tareas de origen aprendidas.

- De forma diferente, la EMO puede utilizarse en escenarios en los que los parámetros de los GPAIS evolucionan conjuntamente dentro de una única búsqueda evolutiva basada en varias funciones objetivo, como se ha hecho en [MDSOH21] en el contexto del aprendizaje por refuerzo multitarea. En este caso, un espacio de búsqueda unificado puede codificar eficazmente los parámetros compartidos por todas las tareas. Los operadores evolutivos adecuados para intercambiar información de los genotipos entre las tareas a optimizar (como los definidos en la optimización multifactorial [BOGT20]) implementan eficazmente la transferencia de conocimientos dichas tareas.

En estas tres áreas de investigación, la transferencia de parámetros, esquemas u otros elementos arquitectónicos pueden acelerar el proceso de aprendizaje y potenciar aún más el rendimiento del modelo. Cuando se trata de GPAIS en estas áreas, el gran espacio de búsqueda de variables de decisión evolucionado por el AE puede suponer un reto. Recientemente, los enfoques de aprendizaje multitarea han estado utilizando AEs multiobjetivo y la optimización global a gran escala para garantizar la escalabilidad para tamaños de modelo realistas [LLF+23]. La evaluación de la transferencia de conocimientos entre tareas a menudo requiere la formación en la tarea o tareas de destino, lo que provoca retrasos significativos. Para reducir este coste, se han utilizado modelos basados en sustitutos en el DL evolutivo [SWX+20]. En GPAIS multitarea, los modelos para diferentes tareas pueden tener distintas latencias de procesamiento, y los modelos basados en sustitutos pueden beneficiarse de los aprendidos para las tareas evaluadas con mayor rapidez [WJSO22].

4.3 EA-GPAIS mediante la creación continua de nuevo conocimiento

4.3.1 ¿Por qué es importante esta estrategia para GPAIS?

Los GPAIS necesitan adaptarse a nuevas tareas con muy pocos datos, por lo que requieren la creación continua de conocimiento diverso para acelerar la capacidad de respuesta y facilitar la integración de tareas. Cuando un GPAIS tiene un buen rendimiento en un conjunto determinado de tareas (escenario de mundo cerrado), su transición a un escenario de mundo abierto se realiza utilizando mecanismos de diversificación del conocimiento, que pueden llevarse a cabo mediante el modelado de conjuntos, la aleatorización de los parámetros del GPAIS o la generación de datos sintéticos, entre otras estrategias.

4.3.2 ¿Cómo pueden contribuir los AEs a esta estrategia?

Los AEs pueden servir para generar modelos diversos y tratar problemas en diferentes espacios de búsqueda. A la hora de aplicar esta estrategia con los AEs, se plantea un doble dilema: (1) cómo formular las funciones objetivo que modelen adecuadamente la diversidad en entornos de mundo abierto, incluso en ausencia de datos de la(s) nueva(s) tarea(s); (2) cómo conservar las soluciones durante la búsqueda evolutiva que sean a la vez buenas para las tareas de origen y diversas de manera que sean potencialmente útiles para acometer las nuevas tareas.

4.3.3 ¿Qué áreas de investigación de los AEs son útiles para esta estrategia?

Diferentes áreas de investigación de los AEs han prestado atención al descubrimiento y la conservación de soluciones diversas durante la búsqueda. Además, los AEs también pueden ser útiles para inducir esta diversidad mediante la síntesis de datos que producen comportamientos diversos del modelo a través del entrenamiento. A continuación revisaremos algunas de estas áreas:

- El *open-ended evolution* se alinea perfectamente con esta estrategia, ya que persigue la generación continua de conocimiento sin llegar a un estado óptimo. La integración del *open-ended evolution* con los AEs hace factible la creación de sistemas que puedan adaptarse continuamente [Tay19], donde los AEs ayudarán a crear más datos o entornos de trabajo [LS11a]. Este es el enfoque seguido en POET [WLCS19] o en *Minimal Criterion Coevolution* [BS17].

- La *quality-diversity optimization* tiene como objetivo desarrollar AEs que prioricen la generación de soluciones diversas y de alta calidad. Además de evaluar la calidad de las soluciones, las métricas de diversidad miden lo distinta que es cada solución respecto a las demás. Promover la diversidad garantiza que las soluciones sean buenas y variadas en todos los comportamientos posibles. Los AEs diseñados según este paradigma son idóneos para la evolución de los GPAIS en entornos de mundo abierto. Mediante la evolución de sus parámetros y/o configuración y el diseño de medidas para cuantificar la diversidad de los GPAIS evolucionados, la *quality-diversity optimization* puede proporcionar un conjunto de modelos diversos y de buen rendimiento. Esto podría ofrecer mejores garantías de adaptabilidad a nuevas tareas, incluso en presencia de objetivos contradictorios [NGWN19].

- La optimización multimodal busca también mantener soluciones diversas y de alta calidad durante la búsqueda evolutiva [Pre15]. Sin embargo, a diferencia de la *quality-diversity optimization*, la diversidad se define sobre el genotipo de las soluciones evolucionadas por el algoritmo en vez de en un espacio de comportamiento. En cualquier caso, la optimización multimodal también puede ser una prometedora área de investigación de AEs para producir modelos diversos, en particular sobre

espacios de búsqueda combinatoria que representan configuraciones de modelos [TI20].

- La *novelty search* implica dirigir la búsqueda en base a una medida de la novedad de las soluciones en el espacio de búsqueda, y no basándose en una función objetivo [LS11b, LS11a]. Esto garantiza el descubrimiento de nuevas soluciones y la acumulación de más conocimiento sobre el problema en cuestión. Aunque tanto la *novelty search* como la *quality-diversity optimization* se esfuerzan por conservar soluciones diversas, la *novelty search* se centra en promover la novedad como medida de la diversidad, mientras que la *quality-diversity optimization* busca un equilibrio entre las soluciones de alta calidad y la diversidad, abarcando múltiples aspectos más allá de la novedad por sí sola. Estas técnicas pueden complementarse para crear nuevo conocimiento para los GPAIS de mundo abierto, y algunos aspectos de la *novelty search* pueden contribuir a la diversidad en el marco de la *quality-diversity optimization*. La *novelty search* con los AEs se ha explorado en robótica y sistemas de conducción autónoma, para garantizar que el conocimiento creado evite el malfuncionamiento del sistema, mostrando las posibilidades que aporta esta área de investigación de los AEs para que los sistemas complejos se adapten a circunstancias desconocidas [LSMC19].

- El *ensemble learning* evolutivo estudia la optimización de los "aprendices" dentro de un *ensemble* de modelos de ML basado en diferentes objetivos. La mayor parte de la literatura se centra en el rendimiento del *ensemble* en tareas de modelado, pero algunos trabajos abordan diferentes objetivos como la igualdad [ZLZ+23] y el manejo de recompensas conflictivas en sistemas de aprendizaje por refuerzo multiagente [BCJ23]. Consideramos que el amplio bagaje de metodologías para modelar la diversidad en el *ensemble learning* evolutivo (recientemente revisado en [Hey24]) puede abrir vías complementarias para realizar GPAIS de mundo abierto basados en *ensembles* de modelos de ML.

4.4 EA-GPAIS mediante la construcción de nuevos modelos desde cero

4.4.1 ¿Por qué es importante esta estrategia para GPAIS?

La construcción de nuevos modelos desde cero es otra estrategia utilizada en los GPAIS. En estos casos, la construcción suele asociarse a los GPAIS de mundo cerrado, ya que requiere la hipótesis de disponer de datos de calidad para formular una función objetivo que guíe la búsqueda del modelo óptimo para la(s) tarea(s) en cuestión. Cuando llegan nuevas tareas, el modelo puede dejar de ser óptimo, por lo que debe optimizarse de nuevo para esas nuevas tareas. Para abordar este problema como un problema de mundo abierto, es crucial modificar cada parte del modelo para las nuevas tareas. Sin embargo, la falta de suficientes datos de alta calidad dificulta esta adaptación. Puede que no haya suficiente conocimiento previo de las nuevas tareas para evaluar con precisión la generalización del

algoritmo desarrollado. En consecuencia, la dificultad reside en adaptar este enfoque a escenarios de mundo abierto.

4.4.2 ¿Cómo pueden contribuir los AEs a esta estrategia?

Los AEs pueden optimizar varias etapas de un modelado de ML, lo que los hace adecuados para los GPAIS. Áreas del ML como AutoML, optimización de hiperparámetros e ingeniería de atributos se han tratado eficazmente mediante AEs [TTBG21]. Los GPAIS pueden heredar este exitoso historial de logros, enfrentándose a nuevos retos como la mayor granularidad a la que se construyen los GPAIS (por ejemplo, primitivas de procesamiento de bajo nivel, como en AutoML-zero [RLSL20]), o la ya mencionada dificultad de formular un objetivo para optimizar sobre escasos datos o incluso ningún conocimiento sobre la(s) nueva(s) tarea(s).

4.4.3 ¿Qué áreas de investigación de los AEs son útiles para esta estrategia?

Áreas como AutoML y la selección de algoritmos ya se han aplicado en GPAIS utilizando AEs [LMH$^+$19]. Además, el AutoML puede identificar etapas de modelado óptimas para enfoques de DL, abarcando la ingeniería de atributos, la optimización de hiperparámetros y NAS. En consecuencia, los AEs han desempeñado un papel importante en la construcción de modelos desde cero en los últimos años. Este compromiso es evidente gracias a los diversos estudios sobre NAS que exploran la construcción de modelos [LSX$^+$23, DSOM$^+$19, DHD20, ZLZ22], y también a los varios GPAIS de mundo cerrado que se encuadran en esta estrategia: AutoML-zero y MOAZ, su extensión a la optimización multiobjetivo. Por último, en el campo de la neuroevolución, CoDeepNEAT [LMH$^+$19, MLM$^+$24a] puede considerarse como una construcción del modelo, a pesar de estar clasificado en la categoría selección de algoritmos. Este enfoque tiene como objetivo identificar las primitivas esenciales para la construcción de una red neuronal. Para ello, se desarrolla la selección y evolución de dos poblaciones: una para las estructuras de la red y otra para sus primitivas.

4.5 Problemas interestratégicos resueltos con los AEs

A continuación nos centramos en el uso de los AEs para afrontar y resolver problemas de múltiples estrategias en el contexto de los GPAIS. En concreto, exploramos cómo los AEs pueden navegar eficazmente por entornos problemáticos y complejos, combinando varias estrategias para adaptarse a diversas condiciones y optimizar soluciones de forma colaborativa:

4.5.1 Escasez de datos de calidad para las tareas objetivo

En el contexto de los GPAIS, el objetivo principal es facilitar la adaptación del sistema ante la aparición de una nueva tarea. Sin embargo, los datos disponibles para esta adaptación pueden ser a menudo insuficientes. Para solucionarlo, se puede recurrir a diversas estrategias, como la síntesis de datos y el *active learning*. La combinación de estas estrategias con los AEs puede reducir el tiempo y mejorar la calidad de la adaptación. Los AEs, como en el caso de POET [WLCS19], no sólo pueden generar datos similares, sino también optimizar grupos de datos nuevos para que el sistema funcione mejor en tareas conocidas, o incluso adaptarse a nuevas tareas de una manera más rápida y eficaz.

4.5.2 Formulación de funciones objetivo en régimen de mundo abierto para *zero-shot learning*

En EA-GPAIS, la cuidadosa selección de las funciones objetivo es crucial para el diseño y la mejora del sistema. Como se ha indicado anteriormente, durante la fase de entrenamiento del GPAIS puede ocurrir que haya un conocimiento incompleto de las tareas objetivo, por lo que se hace necesario preparar al GPAIS para nuevas tareas. Los paradigmas del ML como el *few-shot learning*, el meta-aprendizaje o el aprendizaje por transferencia, que pueden adaptar el modelo para realizar nuevas tareas con un número limitado de ejemplos, se emplean a menudo en escenarios de *zero-shot learning* de mundo abierto. Sin embargo, al tratarlos con EA-GPAIS, resulta crucial formular unos objetivos de optimización que sean capaces de medir la adaptabilidad del GPAIS optimizado a lo desconocido. Las medidas de diversidad, el descubrimiento de tareas auxiliares autosupervisadas que el GPAIS pueda aprender de forma anticipada, o incluso las estrategias de *open-ended evolution* basadas en AEs pueden ser líneas de investigación interesantes para este fin.

4.5.3 Dimensionalidad del espacio de búsqueda

El diseño de un sistema implica tomar decisiones importantes sobre sus parámetros. Cuando se trata de un sistema extremadamente grande, tomar estas decisiones resulta incluso más difícil. En consecuencia, la selección de todos los componentes relevantes durante el diseño de un GPAIS se plantea como una tarea importante. En este contexto, la optimización global a gran escala se perfila como un campo idóneo para afrontar los problemas caracterizados por espacios de alta dimensionalidad. Los AEs han sido ampliamente utilizados en problemas de optimización a gran escala [OLY22]. Mediante el uso de AEs para problemas de alta dimensión y la transferencia de conocimientos para escenarios de mundo abierto, podemos hacer que el diseño o incluso la construcción de grandes GPAIS sea factible y computacionalmente asequible.

4.5.4 Evaluación costosa de las funciones objetivo

La selección de un modelo que defina un sistema dado es una decisión fundamental que influye en el rendimiento del sistema a la hora de realizar una tarea determinada. Cuando esta selección la realiza un AE, la búsqueda evolutiva requiere evaluar la calidad de múltiples modelos candidatos que, dentro del contexto de las tareas de que se realizan en el ML, implican repetidas fases de entrenamiento/validación. Aunque el diseño del modelo, especialmente en el caso de los modelos de mayor tamaño, puede suponer un cuello de botella en el proceso de evaluación, es fundamental señalar que, en el ML, la evaluación de la calidad de dichos modelos también contribuye significativamente al tiempo de evaluación.

Las funciones objetivo que tienen costes computacionales elevados pueden beneficiarse de las ventajas que se buscan adoptando AEs para el diseño y la mejora de los GPAIS. La optimización basada en sustitutos es una solución que permite agilizar los tiempos de evaluación mediante el uso de modelos sustitutos para hacer el proceso más eficiente y explorar así más modelos candidatos. Esto mejora la eficiencia global de los EA-GPAIS. Los AEs y los modelos basados en sustitutos desempeñan un papel fundamental a la hora de la elección del modelo y la eficacia de la evaluación. Estos algoritmos contribuyen a la exploración global dentro del espacio de búsqueda, mientras que los modelos basados en sustitutos mejoran la eficiencia computacional del proceso global, agilizando así el proceso de optimización. Este enfoque colaborativo acelera la convergencia hacia soluciones óptimas, lo que lo convierte en una estrategia convincente para refinar y optimizar los GPAIS.

En conclusión, las áreas de investigación descritas en esta sección delinean un panorama prometedor para futuros estudios sobre el desarrollo de los GPAIS. Estas áreas de investigación basadas en AEs, junto con las que ya contribuyen activamente al avance continuo de los GPAIS, tienen el potencial de fomentar la aparición de sistemas capaces de autoconstruirse y/o autoadaptarse a nuevas tareas, acercándonos así a la IA a una IA más general.

5 Conclusiones

La computación evolutiva desempeña un papel importante en el diseño y la mejora de los sistemas de IA. La integración de los AEs con GPAIS está actualmente dando lugar a avances significativos, como se pone de manifiesto en este estudio. Los objetivos principales de este trabajo han sido: (1) introducir el término EA-GPAIS y una taxonomía para los AEs que potencia a otra IA, (2) analizar las diversas opciones para el diseño y mejora de los GPAIS utilizando la taxonomía de AEs que potencia a otra IA, relacionar las propiedades clave de los GPAIS con las áreas de investigación de AEs que pueden estudiarlas, y (3) estudiar los retos de aprovechar las ventajas de los EA-GPAIS y las estrategias para abordar los avances en EA-GPAIS. Los AEs contribuyen al diseño de los GPAIS optimizando los hiperparámetros y la configuración, seleccionando algoritmos más adecuados o incluso creando algoritmos desde cero. La mejora de los GPAIS con la ayuda de los AEs implica inducir la diversidad mediante mecanismos evolutivos dentro del GPAIS, hacer evolucionar los datos a partir de los cuales aprende el GPAIS u optimizar la transferencia de conocimientos entre tareas.

Existen múltiples propuestas de GPAIS de mundo cerrado en el ámbito del DL, pero es importante reconocer que existen otras áreas que también han influido significativamente en la evolución de este tipo de sistemas. La brecha hacia los GPAIS de mundo abierto se está acortando, como demuestran las propuestas que ejemplifican el uso de la diversidad para generar conocimiento y adaptarse a lo desconocido. Esta noción está inspirando a los investigadores a explorar nuevos horizontes, combinando su experiencia con los AEs para desarrollar los GPAIS de mundo abierto. Con este creciente interés, se prevé un futuro rebosante de modelos EA-GPAIS versátiles que serán referentes en sus respectivos ámbitos.

Desde la perspectiva de la gobernanza de los GPAIS, la inclusión de los AEs en el diseño y mejora de estos sistemas complejos puede aportar ventajas adicionales, ya que proporcionan medios para considerar objetivos de optimización que miden aspectos relacionados con la credibilidad de estos sistemas. Una reflexión general sobre estos temas se ha realizado en [TMP+24].

Como conclusión, la fuerte investigación sobre GPAIS y GPAIS evolutivos que se ha observado en los últimos años pone de manifiesto las grandes expectativas depositadas en este campo a la hora de revolucionar el campo del ML evolutivo, dando lugar a sistemas de IA más generales capaces de resolver un amplio abanico de tareas y de adaptar por sí mismos sus conocimientos para abordar otras nuevas. La flexibilidad y facilidad de adaptación de los AEs los convierte en un complemento perfecto para hacer frente a las estrictas propiedades que se buscan en los GPAIS, incluyendo la multimodalidad de las tareas a resolver, su variabilidad en el tiempo, o la gran dimensionalidad del diseño y de su construcción. La hibridación de esta familia de algoritmos con los GPAIS promete un futuro brillante en el campo de la IA, que representará la vanguardia de la investigación en los próximos años con el objetivo fundamental de desarrollar sistemas similares a los humanos, capaces de analizar, aprender y realizar diversas tareas por sí mismos.

Bibliography

[ABBS17] Azzouz R., Bechikh S., and Ben Said L. (2017) Dynamic Multi-objective Optimization Using Evolutionary Algorithms: A Survey. In *Recent Advances in Evolutionary Multi-objective Optimization*, pp. 31–70. Springer International Publishing.

[ALMR19] Assunçao F., Lourenço N., Machado P., and Ribeiro B. (2019) DENSER: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines* 20: 5–35.

[ASBC+19] Al-Sahaf H., Bi Y., Chen Q., Lensen A., Mei Y., Sun Y., Tran B., Xue B., and Zhang M. (2019) A survey on evolutionary machine learning. *Journal of the Royal Society of New Zealand* 49(2): 205–228.

[AST+24] Akiba T., Shing M., Tang Y., Sun Q., and Ha D. (2024) Evolutionary Optimization of Model Merging Recipes. arXiv:2403.13187.

[BBdCF12] Barros R. C., Basgalupp M. P., de Carvalho A. C. P. L. F., and Freitas A. A. (2012) A Survey of Evolutionary Algorithms for Decision-Tree Induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42(3): 291–312.

[BBS20] Bharti V., Biswas B., and Shukla K. K. (2020) Recent Trends in Nature Inspired Computation with Applications to Deep Learning. In *10th International Conference on Cloud Computing, Data Science & Engineering*, pp. 294–299.

[BCJ23] Bai H., Cheng R., and Jin Y. (2023) Evolutionary Reinforcement Learning: A Survey. *Intelligent Computing* 2: 0025.

[BDT+23] Bradley H., Dai A., Teufel H. B., Zhang J., Oostermeijer K., Bellagente M., Clune J., Stanley K., Schott G., and Lehman J. (2023) Quality-Diversity through AI Feedback. In *Second Agent Learning in Open-Endedness Workshop*, pp. 1–112.

[BFM97] Back T., Fogel D. B., and Michalewicz Z. (1997) *Handbook of Evolutionary Computation*. IOP Publishing Ltd.

[BHA+22] Bommasani R., Hudson D. A., Adeli E., Altman R., Arora S., von Arx S., Bernstein M. S., Bohg J., Bosselut A., Brunskill E., *et al.* (2022) On the Opportunities and Risks of Foundation Models. arXiv:2108.07258.

[BKvS+23] Bäck T. H., Kononova A. V., van Stein B., Wang H., Antonov K. A., Kalkreuth R. T., de Nobel J., Vermetten D., de Winter R., and Ye F. (2023) Evolutionary Algorithms for Parameter Optimization—Thirty Years Later. *Evolutionary Computation* 31(2): 81–122.

[BNN+23] Barrett A. M., Newman J., Nonnecke B., Hendrycks D., Murphy E. R., and Jackson K. (2023) AI risk-management standards profile for general-purpose AI systems (GPAIS) and foundation models. Technical report, Center for Long-Term Cybersecurity.

[BOGT20] Bali K. K., Ong Y.-S., Gupta A., and Tan P. S. (2020) Multifactorial Evolutionary Algorithm With Online Transfer Parameter Estimation: MFEA-II. *IEEE Transactions on Evolutionary Computation* 24(1): 69–83.

[BS17] Brant J. C. and Stanley K. O. (2017) Minimal Criterion Coevolution: A New Approach to Open-Ended Search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 67–74.

[CCH+23] Chen S., Chen S., Hou W., Ding W., and You X. (2023) EGANS: Evolutionary Generative Adversarial Network Search for Zero-Shot Learning. *Early Access in IEEE Transactions on Evolutionary Computation* .

[CD18] Cully A. and Demiris Y. (2018) Quality and Diversity Optimization: A Unifying Modular Framework. *IEEE Transactions on Evolutionary Computation* 22(2): 245–259.

[CL23] Campos S. and Laurent R. (2023) A Definition of General-Purpose AI Systems: Mitigating Risks from the Most Generally Capable Models. *Available at SSRN* 4423706: 1–8.

[CLVV+14] Coello C. A. C., Lamont G. B., Van Veldhuizen D. A., *et al.* (2014) *Evolutionary algorithms for solving multi-objective problems*, volumen 5. Springer.

[CRMdJ20] Charte F., Rivera A. J., Martínez F., and del Jesus M. J. (2020) EvoAAA: An evolutionary methodology for automated neural autoencoder architecture search. *Integrated Computer-Aided Engineering* 27(3): 211–231.

[CVSD20] Camacho Villalón C. L., Stützle T., and Dorigo M. (2020) Grey Wolf, Firefly and Bat Algorithms: Three Widespread Algorithms that Do Not Contain Any Novelty. In *International Conference on Swarm Intelligence*, volumen 12421, pp. 121–133.

[CWM12] Chiong R., Weise T., and Michalewicz Z. (2012) *Variants of evolutionary algorithms for real-world applications*, volumen 2. Springer.

[DB17] Dufourq E. and Bassett B. A. (2017) EDEN: Evolutionary deep networks for efficient machine learning. In *Pattern Recognition Association of South Africa and Robotics and Mechatronics*, pp. 110–115.

[DFDF+18] Di Francescomarino C., Dumas M., Federici M., Ghidini C., Maggi F. M., Rizzi W., and Simonetto L. (2018) Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems* 74: 67–83.

[DHD20] Darwish A., Hassanien A. E., and Das S. (2020) A survey of swarm and evolutionary computing approaches for deep learning. *Artificial Intelligence Review* 53: 1767–1812.

[DSOM+19] Del Ser J., Osaba E., Molina D., Yang X.-S., Salcedo-Sanz S., Camacho D., Das S., Suganthan P. N., Coello C. A. C., and Herrera F. (Agosto 2019) Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation* 48: 220–250.

[EMH19a] Elsken T., Metzen J. H., and Hutter F. (2019) Efficient Multi-Objective Neural Architecture Search via Lamarckian Evolution. In *International Conference on Learning Representations*, pp. 1–23.

[EMH19b] Elsken T., Metzen J. H., and Hutter F. (2019) Neural Architecture Search: A Survey. *Journal of Machine Learning Research* 20(55): 1–21.

[EVH10] Espejo P. G., Ventura S., and Herrera F. (2010) A Survey on the Application of Genetic Programming to Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).* 40(2): 121–144.

[FLSL19] Fu S., Li Y., Sun S., and Li H. (2019) Evolutionary support vector machine for RMB exchange rate forecasting. *Physica A: Statistical Mechanics and its Applications* 521: 692–704.

[FYHT17] Fang M., Yin J., Hall L. O., and Tao D. (2017) Active Multitask Learning With Trace Norm Regularization Based on Excess Risk. *IEEE Transactions on Cybernetics* 47(11): 3906–3915.

[GAK+23] Guha R., Ao W., Kelly S., Boddeti V., Goodman E., Banzhaf W., and Deb K. (2023) MOAZ: A Multi-Objective AutoML-Zero Framework. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 485–492.

[GAU+23] Gutierrez C. I., Aguirre A., Uuk R., Boine C. C., and Franklin M. (2023) A Proposal for a Definition of General Purpose Artificial Intelligence Systems. *Digital Society* 2(36): 1–8.

[GBC16] Goodfellow I., Bengio Y., and Courville A. (2016) *Deep learning.* MIT press.

[GL23] Gen M. and Lin L. (2023) Genetic Algorithms and Their Applications. In *Springer Handbook of Engineering Statistics*, pp. 635–674. Springer London, London.

[GWG+24] Guo Q., Wang R., Guo J., Li B., Song K., Tan X., Liu G., Bian J., and Yang Y. (2024) Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers. In *The Twelfth International Conference on Learning Representations*.

[Hey24] Heywood M. I. (2024) Evolutionary Ensemble Learning. In *Handbook of Evolutionary Machine Learning*, pp. 205–243. Springer Nature Singapore, Singapore.

[HKV19] Hutter F., Kotthoff L., and Vanschoren J. (2019) *Automated Machine Learning - Methods, Systems, Challenges*. Springer.

[HM15] Hirschberg J. and Manning C. D. (2015) Advances in natural language processing. *Science* 349(6245): 261–266.

[HMO24] Hemberg E., Moskal S., and O'Reilly U.-M. (2024) Evolving Code with A Large Language Model. arXiv:2401.07102.

[JWQ+21] Jiang M., Wang Z., Qiu L., Guo S., Gao X., and Tan K. C. (2021) A Fast Dynamic Evolutionary Multiobjective Algorithm via Manifold Transfer Learning. *IEEE Transactions on Cybernetics* 51(7): 3417–3428.

[KGJH16] Krawczyk B., Galar M., Jelen L., and Herrera F. (2016) Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. *Applied Soft Computing* 38: 714–726.

[LDG+20] Lu Z., Deb K., Goodman E., Banzhaf W., and Boddeti V. N. (2020) NSGANetV2: Evolutionary Multi-objective Surrogate-Assisted Neural Architecture Search. In *European Conference on Computer Vision*, pp. 35–51.

[LGJ+24] Lehman J., Gordon J., Jain S., Ndousse K., Yeh C., and Stanley K. O. (2024) Evolution Through Large Models. In *Handbook of Evolutionary Machine Learning*, pp. 331–366. Springer Nature Singapore, Singapore.

[LJJ22] Luo R., Ji S., and Ji Y. (2022) An active-learning Pareto evolutionary algorithm for parcel locker network design considering accessibility of customers. *Computers & Operations Research* 141: 105677.

[LLF+23] Liu S., Lin Q., Feng L., Wong K.-C., and Tan K. C. (2023) Evolutionary Multitasking for Large-Scale Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation* 27(4): 863–877.

[LLL14] Le Ly D. and Lipson H. (2014) Optimal Experiment Design for Coevolutionary Active Learning. *IEEE Transactions on Evolutionary Computation* 18(3): 394–404.

[LLS18] Liang S., Li Y., and Srikant R. (2018) Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proceedings of the 6th International Conference on Learning Representations*.

[LMH+19] Liang J., Meyerson E., Hodjat B., Fink D., Mutch K., and Miikkulainen R. (2019) Evolutionary Neural AutoML for Deep Learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 401–409.

[LMX+23] Li N., Ma L., Xing T., Yu G., Wang C., Wen Y., Cheng S., and Gao S. (2023) Automatic design of machine learning via evolutionary computation: A survey. *Applied Soft Compuing* 143: 110412.

[LMY+23] Li N., Ma L., Yu G., Xue B., Zhang M., and Jin Y. (2023) Survey on Evolutionary Deep Learning: Principles, Algorithms, Applications, and Open Issues. *ACM Computing Surveys* 56(2): 1–34.

[LS08] Lehman J. and Stanley K. O. (2008) Exploiting open-endedness to solve problems through the search for novelty. In *Artificial Life XI: Proceedings of the 11th International Conference on the Simulation and Synthesis of Living Systems*, pp. 329–336.

[LS11a] Lehman J. and Stanley K. O. (2011) Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation* 19(2): 189–223.

[LS11b] Lehman J. and Stanley K. O. (2011) Novelty Search and the Problem with Objectives. In *Genetic Programming Theory and Practice IX*, pp. 37–56. Springer New York, New York.

[LSG+21] Lu Z., Sreekumar G., Goodman E., Banzhaf W., Deb K., and Boddeti V. N. (2021) Neural Architecture Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43(9): 2971–2989.

[LSMC19] Langford M. A., Simon G. A., McKinley P. K., and Cheng B. H. C. (2019) Applying Evolution and Novelty Search to Enhance the Resilience of Autonomous Systems. In *IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 63–69.

[LSX+23] Liu Y., Sun Y., Xue B., Zhang M., Yen G. G., and Tan K. C. (2023) A Survey on Evolutionary Neural Architecture Search. *IEEE Transactions on Neural Networks and Learning Systems* 34(2): 550–570.

[LTYZ23] Liu F., Tong X., Yuan M., and Zhang Q. (2023) Algorithm Evolution Using Large Language Model. arXiv:2311.15249.

[LWB+19] Lu Z., Whalen I., Boddeti V., Dhebar Y., Deb K., Goodman E., and Banzhaf W. (2019) NSGA-Net: Neural Architecture Search Using Multi-Objective Genetic Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 419–427.

[MDSOH21] Martinez A. D., Del Ser J., Osaba E., and Herrera F. (2021) Adaptive Multifactorial Evolutionary Optimization for Multitask Reinforcement Learning. *IEEE Transactions on Evolutionary Computation* 26(2): 233–247.

[MDVR+21] Martinez A. D., Del Ser J., Villar-Rodriguez E., Osaba E., Poyatos J., Tabik S., Molina D., and Herrera F. (2021) Lights and shadows in Evolutionary Deep Learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges. *Information Fusion* 67: 161–194.

[Mii24] Miikkulainen R. (2024) Generative AI: An AI paradigm shift in the making? *AI Magazine* 45: 165–167.

[MLM+24a] Miikkulainen R., Liang J., Meyerson E., Rawal A., Fink D., Francon O., Raju B., Shahrzad H., Navruzyan A., Duffy N., *et al.* (2024) Evolving Deep Neural Networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing (Second Edition)*, chapter 14, pp. 269–287. Academic Press.

[MLM+24b] Miikkulainen R., Liang J., Meyerson E., Rawal A., Fink D., Francon O., Raju B., Shahrzad H., Navruzyan A., Duffy N., and Hodjat B. (2024) Evolving Deep Neural Networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, chapter 15, pp. 293–312. Academic Press.

[MLW+23] Ma Y. J., Liang W., Wang G., Huang D.-A., Bastani O., Jayaraman D., Zhu Y., Fan L., and Anandkumar A. (2023) Eureka: Human-level reward design via coding large language models. arXiv:2310.12931.

[MLZ+19] Ma X., Li X., Zhang Q., Tang K., Liang Z., Xie W., and Zhu Z. (2019) A Survey on Cooperative Co-Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 23(3): 421–441.

[MSC+13] Mikolov T., Sutskever I., Chen K., Corrado G. S., and Dean J. (2013) Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*, volumen 26, page 3111–3119.

[MSNM21] Mauceri S., Sweeney J., Nicolau M., and McDermott J. (2021) Feature extraction by grammatical evolution for one-class time series classification. *Genetic Programming and Evolvable Machines* 22(3): 267–295.

[NGWN19] Neumann A., Gao W., Wagner M., and Neumann F. (2019) Evolutionary Diversity Optimization Using Multi-Objective Indicators. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 837–845.

[NYB12] Nguyen T. T., Yang S., and Branke J. (2012) Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation* 6: 1–24.

[ODSMH22] Osaba E., Del Ser J., Martinez A. D., and Hussain A. (2022) Evolutionary multitask optimization: a methodological overview, challenges, and future research directions. *Cognitive Computation* 14(3): 927–954.

[OLY22] Omidvar M. N., Li X., and Yao X. (2022) A Review of Population-Based Metaheuristics for Large-Scale Black-Box Global Optimization—part I. *IEEE Transactions on Evolutionary Computation* 26(5): 802–822.

[OSB+14] Orive D., Sorrosal G., Borges C. E., Martin C., and Alonso-Vicario A. (2014) Evolutionary algorithms for hyperparameter tuning on neural networks models. In *Proceedings of the 26th European Modeling & Simulation Symposium*, pp. 402–409.

[PBC+19] Packard N., Bedau M. A., Channon A., Ikegami T., Rasmussen S., Stanley K. O., and Taylor T. (2019) An Overview of Open-Ended Evolution: Editorial Introduction to the Open-Ended Evolution II Special Issue. *Artificial Life* 25(2): 93–103.

[PKP+19] Parisi G. I., Kemker R., Part J. L., Kanan C., and Wermter S. (2019) Continual lifelong learning with neural networks: A review. *Neural Networks* 113: 54–71.

[PL23] Phan Q. M. and Luong N. H. (2023) Enhancing multi-objective evolutionary neural architecture search with training-free Pareto local search. *Applied Intelligence* 53(8): 8654–8672.

[PMMS+23] Poyatos J., Molina D., Martínez-Seras A., Del Ser J., and Herrera F. (2023) Multiobjective evolutionary pruning of Deep Neural Networks with Transfer Learning for improving their performance and robustness. *Applied Soft Computing* 147: 110757.

[Pre15] Preuss M. (2015) *Multimodal optimization by means of evolutionary algorithms*. Springer.

[PSS16] Pugh J. K., Soros L. B., and Stanley K. O. (2016) Quality Diversity: A New Frontier for Evolutionary Computation. *Frontiers in Robotics and AI* 3: 1–40.

[PY09] Pan S. J. and Yang Q. (2009) A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10): 1345–1359.

[RLSL20] Real E., Liang C., So D., and Le Q. (2020) AutoML-Zero: Evolving Machine Learning Algorithms From Scratch. In *Proceedings of the 37th International Conference on Machine Learning*, volumen 119, pp. 8007–8019.

[RMS+17] Real E., Moore S., Selle A., Saxena S., Suematsu Y. L., Tan J., Le Q. V., and Kurakin A. (2017) Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, volumen 70, pp. 2902–2911.

[RV18] Reyes O. and Ventura S. (2018) Evolutionary Strategy to Perform Batch-Mode Active Learning on Multi-Label Data. *ACM Transactions on Intelligent Systems and Technology* 9(4): 1–26.

[RW17] Rawat W. and Wang Z. (2017) Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation* 29(9): 2352–2449.

[SCLM19] Stanley K. O., Clune J., Lehman J., and Miikkulainen R. (2019) Designing neural networks through neuroevolution. *Nature Machine Intelligence* 1(1): 24–35.

[SCZZ20] Sun S., Cao Z., Zhu H., and Zhao J. (2020) A Survey of Optimization Methods From a Machine Learning Perspective. *IEEE Transactions on Cybernetics* 50(8): 3668–3681.

[SEBB20] Stork J., Eiben A., and Bartz-Beielstein T. (2020) A new taxonomy of global optimization algorithms. *Natural Computing* 21: 219–242.

[Sim13] Simon D. (2013) *Evolutionary optimization algorithms*. John Wiley & Sons.

[SM02] Stanley K. O. and Miikkulainen R. (2002) Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10(2): 99–127.

[Sta03] Standish R. K. (2003) Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications* 3(02): 167–175.

[STÖ19] Song H., Triguero I., and Özcan E. (2019) A review on the self and dual interactions between machine learning and optimisation. *Progress in Artificial Intelligence* 8(2): 143–165.

[SWVN23] Stokel-Walker C. and Van Noorden R. (2023) What ChatGPT and generative AI mean for science. *Nature* 614(7947): 214–216.

[SWX+20] Sun Y., Wang H., Xue B., Jin Y., Yen G. G., and Zhang M. (2020) Surrogate-Assisted Evolutionary Deep Learning Using an End-to-End Random Forest-Based Performance Predictor. *IEEE Transactions on Evolutionary Computation* 24(2): 350–364.

[SXZ+20] Sun Y., Xue B., Zhang M., Yen G. G., and Lv J. (2020) Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification. *IEEE Transactions on Cybernetics* 50(9): 3840–3854.

[SXZY20] Sun Y., Xue B., Zhang M., and Yen G. G. (2020) Evolving Deep Convolutional Neural Networks for Image Classification. *IEEE Transactions on Evolutionary Computation* 24(2): 394–407.

[Tay19] Taylor T. (2019) Evolutionary Innovations and Where to Find Them: Routes to Open-Ended Evolution in Natural and Artificial Systems. *Artificial Life* 25(2): 207–224.

[TBC+16] Taylor T., Bedau M., Channon A., Ackley D., Banzhaf W., Beslon G., Dolson E., Froese T., Hickinbotham S., Ikegami T., *et al.* (2016) Open-Ended Evolution: Perspectives from the OEE Workshop in York. *Artificial Life* 22(3): 408–423.

[TD21] Tzanetos A. and Dounias G. (2021) Nature inspired optimization algorithms or simply variations of metaheuristics? *Artificial Intelligence Review* 54(3): 1841–1862.

[TFJ21] Tan K. C., Feng L., and Jiang M. (2021) Evolutionary Transfer Optimization - A New Frontier in Evolutionary Computation Research. *IEEE Computational Intelligence Magazine* 16(1): 22–33.

[TI20] Tanabe R. and Ishibuchi H. (2020) A Review of Evolutionary Multimodal Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation* 24(1): 193–200.

[TMP+24] Triguero I., Molina D., Poyatos J., Del Ser J., and Herrera F. (2024) General Purpose Artificial Intelligence Systems (GPAIS): Properties, definition, taxonomy, societal implications and responsible governance. *Information Fusion* 103: 102135.

[TSK+18] Tan C., Sun F., Kong T., Zhang W., Yang C., and Liu C. (2018) A Survey on Deep Transfer Learning. In *Artificial Neural Networks and Machine Learning*, pp. 270–279.

[TTBG21] Telikani A., Tahmassebi A., Banzhaf W., and Gandomi A. H. (2021) Evolutionary Machine Learning: A Survey. *ACM Computing Surveys* 54(8): 1–35.

[TXZ16] Tran B., Xue B., and Zhang M. (2016) Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing* 8: 3–15.

[ÜB22] Ünal H. T. and Başçiftçi F. (2022) Evolutionary design of neural network architectures: a review of three decades of research. *Artificial Intelligence Review* 55: 1723–1802.

[UGT23] Uuk R., Gutierrez C. I., and Tamkin A. (2023) Operationalising the Definition of General Purpose AI Systems: Assessing Four Approaches. *Available at SSRN* 4471151: 1–14.

[VDBZ+23] Van Dis E. A., Bollen J., Zuidema W., Van Rooij R., and Bockting C. L. (2023) ChatGPT: five priorities for research. *Nature* 614(7947): 224–226.

[Wis20] Wistuba M. (2020) XferNAS: Transfer Neural Architecture Search. In *Proceedings of Machine Learning and Knowledge Discovery in Databases: European Conference*, page 247–262.

[WJSO22] Wang X., Jin Y., Schmitt S., and Olhofer M. (2022) Transfer Learning Based Co-Surrogate Assisted Evolutionary Bi-Objective Optimization for Objectives with Non-Uniform Evaluation times. *Evolutionary Computation* 30(2): 221–251.

[WLCS19] Wang R., Lehman J., Clune J., and Stanley K. O. (2019) Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. arXiv:1901.01753.

[WRH17] Wang Y.-X., Ramanan D., and Hebert M. (2017) Learning to Model the Tail. In *Advances in Neural Information Processing Systems*, volumen 30, page 7032–7042.

[WWW+24] Wu X., Wu S.-h., Wu J., Feng L., and Tan K. C. (2024) Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap. arXiv:2401.10034.

[WWZ+23] Wu L., Wu D., Zhao T., Cai X., and Xie L. (2023) Dynamic multi-objective evolutionary algorithm based on knowledge transfer. *Information Sciences* 636: 118886.

[WYKN20] Wang Y., Yao Q., Kwok J. T., and Ni L. M. (2020) Generalizing from a Few Examples: A Survey on Few-Shot Learning. *ACM Computing Surveys* 53(3): 1–34.

[XQX22] Xu H., Qin A. K., and Xia S. (2022) Evolutionary Multitask Optimization With Adaptive Knowledge Transfer. *IEEE Transactions on Evolutionary Computation* 26(2): 290–303.

[XZBY16] Xue B., Zhang M., Browne W. N., and Yao X. (2016) A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation* 20(4): 606–626.

[Yan10] Yang X.-S. (2010) *Engineering optimization: an introduction with metaheuristic applications.* John Wiley & Sons.

[YOY+23] Yazdani D., Omidvar M. N., Yazdani D., Branke J., Nguyen T. T., Gandomi A. H., Jin Y., and Yao X. (2023) Robust Optimization Over Time: A Critical Review. *IEEE Transactions on Evolutionary Computation* In Press, 10.1109/TEVC.2023.3306017.

[YS20] Yang L. and Shami A. (2020) On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415: 295–316.

[YZ20] Yu T. and Zhu H. (2020) Hyper-parameter optimization: A review of algorithms and applications. arXiv:2003.05689.

[ZLSC23] Zhang J., Lehman J., Stanley K., and Clune J. (2023) OMNI: Open-endedness via Models of human Notions of Interestingness. arXiv:2306.01711.

[ZLZ22] Zhan Z.-H., Li J.-Y., and Zhang J. (2022) Evolutionary deep learning: A survey. *Neurocomputing* 483: 42–58.

[ZLZ+23] Zhang Q., Liu J., Zhang Z., Wen J., Mao B., and Yao X. (2023) Mitigating Unfairness via Evolutionary Multiobjective Ensemble Learning. *IEEE Transactions on Evolutionary Computation* 27(4): 848–862.

[ZNL+23] Zhao H., Ning X., Liu X., Wang C., and Liu J. (2023) What makes evolutionary multi-task optimization better: A comprehensive survey. *Applied Soft Computing* 145: 110545.

[ZQD+21] Zhuang F., Qi Z., Duan K., Xi D., Zhu Y., Zhu H., Xiong H., and He Q. (2021) A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE* 109(1): 43–76.

[ZQGT21] Zhou X., Qin A. K., Gong M., and Tan K. C. (2021) A survey on Evolutionary Construction of Deep Neural Networks. *IEEE Transactions on Evolutionary Computation* 25(5): 894–912.