University of Granada

Department of Computer Science and Artificial Intelligence

PhD Program in Information and Communication Technologies

EVOLUTIONARY COMPUTATION FOR MULTITASK AND META REINFORCEMENT LEARNING: NEW METHODS AND PERSPECTIVES TOWARDS GENERAL-PURPOSE ARTIFICIAL INTELLIGENCE

PhD STUDENT

ARITZ DAVID MARTÍNEZ QUINTANA

PhD ADVISORS

JAVIER DEL SER LORENTE

FRANCISCO HERRERA TRIGUERO

Granada, February 2023

Editor: Universidad de Granada. Tesis Doctorales
Autor: Aritz David Martínez Quintana
ISBN: 978-84-1117-848-8
URI: https://hdl.handle.net/10481/82008

AUTORIZACIÓN / AUTHORIZATION

El doctorando / The doctoral candidate Aritz David Martínez Quintana y los directores de la tesis / and the thesis supervisors: Javier Del Ser Lorente and Francisco Herrera Triguero :

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 supervisor/s and, as far as our knowledge reaches, in the performance of the work, the rights of other authors to be cited (when their results or publications have been used) have been respected.

Granada, February 2023

Doctorando / Doctoral candidate:

Sgd.: Aritz David Martínez Quintana

Director de la tesis / Thesis supervisor: Director de la tesis / Thesis supervisor:

Sgd.: Javier Del Ser Lorente Francisco Herrera Triguero

To my family.
In memory of those we've lost.
I love you.

ACKNOWLEDGEMENTS

At this point, it is easy to look back and see everything lived in retrospect. These have been tough years, full of ups and downs, and now the time has come to close the cycle. I am absolutely grateful to Prof. Dr. Javier (*Javi*) Del Ser, who has guided and helped me from the beginning. Thank you for guiding me through the tangled path of science and for being always supportive. You are a model to follow and I would dare to say that I am as proud of you as you can be of me. I wish to thank Prof. Dr. Francisco Herrera and University of Granada for giving me the chance to work with other top-level researchers.

I would also like to thank Tecnalia for providing me with the resources to carry out this thesis. Also, I am very grateful to all people in Tecnalia who have patiently helped and supported me throughout these years, and for their friendship outside and inside the team.

Finally, I would like to thank my family, especially my parents for having given everything for me, for giving me an education, and teaching me the most important values in life. To my mother, special thanks for believing and feeling my victories as your own, there is no better feeling than seeing how proud you are of what we have achieved.

RESUMEN

En la actualidad, las técnicas de Big Data y aprendizaje profundo (*Deep Learning*) están cambiando la forma en la que interactuamos con la tecnología. Desde recomendadores de contenido hasta tecnologías capaces de crear arte, la ubucuidad de las redes neuronales es evidente hoy día, y se preveé creciente en el medio/largo plazo. Por ello, y ante la inmensidad de campos en los que el *Deep Learning* es aplicable, resulta interesante extrapolar o "reutilizar" el conocimiento generado en un problema para resolver otros problemas relacionados con mayor eficacia y rapidez. Este procedimiento, conocido como aprendizaje por transferencia (*Transfer Learning*), es una técnica muy extendida en Deep Learning. En este sentido, un paradigma del aprendizaje en el que la transferencia de conocimiento entre problemas ha demostrado ser muy efectiva es el aprendizaje por refuerzo (*Reinforcement Learning*), ya que atiende varias de las debilidades inherentes al proceso de entrenamiento de un agente: la eficiencia de muestreo en la exploración del espacio de soluciones, o la posibilidad de que el entrenamiento del agente se estanque en políticas sub-óptimas. Además de las técnicas tradicionalmente empleadas para paliar estos inconvenientes, como la utilización de múltiples agentes o el uso de mecanismos de inducción de curiosidad comportamental, se ha demostrado que la computación evolutiva puede dar lugar a procedimientos híbridos de entrenamiento eficientes en tiempo para agentes de aprendizaje por refuerzo en entornos de aplicación complejos.

En este contexto, la presente tesis doctoral estudia cómo la computación evolutiva puede ayudar a que los modelos de aprendizaje por refuerzo basados en *Deep Learning* sean capaces de adaptarse rápidamente a nuevos escenarios merced a la reutilización del conocimiento generado en problemas precedentes. Para ello, la investigación se centrará en el uso de una rama concreta de reciente aparición en la computación evolutiva, denominados algoritmos multifactoriales, que permiten resolver varios problemas de optimización de manera simultánea, aprovechando las posibles sinergias existentes entre sus espacios de búsqueda y/o soluciones. La tesis parte de la observación de que el entrenamiento de un modelo de aprendizaje por refuerzo basado en *Deep Learning* puede ser formulado como un problema de optimización y por tanto, abordable mediante computación evolutiva. Esta observación abre la posibilidad de que, en escenarios de aprendizaje por refuerzo con múltiples tareas (*multitask reinforcement learning*), los algoritmos multifactoriales anteriormente citados puedan ser empleados para automatizar el intercambio de conocimiento modelado para cada una de las tareas entre los agentes que atacan cada una de ellas.

Esta primera hipótesis de investigación abordada por la tesis se complementa con una segunda idea: la generación de conocimiento generalizable a nuevas tareas de aprendizaje por refuerzo a partir del entrenamiento conjunto de agentes en otras tareas previas. En particular la tesis se centra en la casuística *zero-shot*, por la que no es posible conocer a priori nada de las nuevas tareas, ni actualizar el modelo a posteriori con información recolectada de dichas tareas. Este escenario, también abordado mediante computación evolutiva y algoritmos multifactoriales, supone un paso más allá hacia la capacidad de los modelos de Inteligencia Artificial para generar conocimiento generalizable que le permita adaptarse autónoma y eficientemente a nuevas tareas de aprendizaje, avanzando firmemente hacia un nuevo paradigma del aprendizaje: GPAI (*General-Purpose Artificial Intelligence*).

ABSTRACT

Currently, Big Data techniques and Deep Learning are changing the way humankind interacts with technology. From content recommendation to technologies capable of creating art, the ubiquity of neural networks is evident today, and is expected to grow in the medium to long term. Given the diversity of fields where Deep Learning is applied nowadays, it is interesting to extrapolate or "reuse" the knowledge generated in one problem to solve other related problems with proficiency, efficiency and speed. This procedure, known as *Transfer Learning*, is widely used in modeling tasks resorting to Deep Learning models. In this sense, a paradigm in which knowledge transfer between tasks has been shown to be very effective is Reinforcement Learning. Indeed, Transfer Learning addresses several of the inherent weaknesses in the learning process of an agent: the sampling efficiency when exploring the environment to be solved, or the possibility that the agent's training may get stuck in sub-optimal policies. Besides traditionally used techniques to alleviate these drawbacks, such as the use of multiple agents or mechanisms to induce behavioral curiosity, it has been shown that evolutionary computation can give rise to efficient hybrid training procedures for developing reinforcement learning agents suited to deal with challenging environments.

In this context, this Thesis studies how evolutionary computation can help Reinforcement Learning models based on Deep Learning to quickly adapt to new scenarios through the reuse of knowledge generated in previous modeling problems. For this purpose, the research focus is placed on the use of a specific branch of recently appeared in evolutionary computation, known as multi-factorial algorithms. Techniques belonging to this family of evolutionary optimization methods allow solving several problem instances simultaneously, taking advantage of possible synergies existing between their search space and/or solutions. The Thesis departs from the observation that the training process of a Reinforcement Learning model based on Deep Learning can be formulated as an optimization problem, and therefore, is feasible to be tackled by using evolutionary computation. This observation paves the way towards the possibility that, in multitask Reinforcement Learning scenarios, the previously mentioned multi-factorial algorithms can be used to automate the exchange of knowledge modeled for each of the tasks among the agents addressing each of such tasks.

This first research hypothesis addressed by the Thesis is complemented by a second idea: the generation of generalizable knowledge to new Reinforcement Learning tasks from the simultaneous training of agents on previous Reinforcement Learning tasks. In particular, the Thesis focuses on the zero-shot assumption, by which it is not possible to know beforehand

anything about the new tasks to be addressed, nor to update the model with information collected from these tasks during inference time. This scenario, also tackled through evolutionary computation and multi-factorial algorithms, represents a step forward towards the ability of Artificial Intelligence models to generate knowledge that allows them to adapt autonomously and efficiently to new tasks, advancing steadily towards a new paradigm: GPAI (*General-Purpose Artificial Intelligence*).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

Deep Learning

DL	Deep Learning
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
GAN	Generative Adversarial Network
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
LSTM	Long Short-Term Memory
RL	Reinforcement Learning
TL	Transfer Learning
AE	Auto-Encoder
ESN	Echo State Network
SGD	Stochastic Gradient Descent
DAG	Directed Acyclic Graph
MAML	Model Agnostic Meta-learning
PEARL	Probabilistic Embeddings for Actor-critic meta-RL
GPAI	General Purpose Artificial Intelligence

Meta-Heuristics

EA	Evolutionary Algorithm
TS	Tabu Search
SA	Simulated Annealing
GA	Genetic Algorithm
FA	Firefly Algorithm
HS	Harmony Search
CS	Cuckoo search
BA	Bat Algorithm
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization

ABC	Artificial Bee Colony
DE	Differential Evolution
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CGP	Cartesian Genetic Programming
NS	Novelty Search
DSGE	Dynamic Structured Grammatical Evolution
ES	Evolution Strategy
SHADE-ILS	SHADE with Iterative Local Search
EDA	Estimation of Distribution Algorithm

Transfer Optimization

TO	Transfer Optimization
ETO	Evolutionary Transfer Optimization
EM	Evolutionary Multitasking
MFO	Multifactorial Optimization
MM	Multipopulation-based Multitasking
MFO	Multi Factorial Optimization
MFEA	Multi Factorial Evolutionary Algorithm
MO-MFEA	Multi-Objective MFEA
A-MFEA-RL	Adaptive MFEA for RL
AML-MFEA	Adaptive Meta-Learning MFEA

Evolutionary Deep Learning

EDL	Evolutionary Deep Learning
NE	NeuroEvolution
NEAT	NeuroEvolution of Augmenting Topologies
NAS	Neural Architecture Search
CKA	Centered Kernel Alignment

Part I

INTRODUCTION

1 PRELIMINARIES, MOTIVATION AND OBJECTIVES

"It's dangerous to go alone, take this!"
– The Legend of Zelda

1.1. Context: Deep Learning and Knowledge Transfer

Nowadays there is overall consensus on the capital importance gained by Deep Learning (DL) in the Artificial Intelligence field [1]. Initial results of DL date back to the late 80's, stepping on a history of preceding achievements in neural computation [2]. However, it was not until years later when advances in high-performance computing, new achievements in neural network training [3], and the availability of massive datasets paved the way for the renowned success of this family of learning models. Nowadays, plenty of application areas have harnessed the superior modeling capabilities of DL models, including natural language processing [4], speech and audio processing [5, 6], social network analysis [7] or autonomous driving [8], to mention a few. As a result, DL models such as Convolutional Neural Networks (CNNs) [9], Recurrent Neural Networks (RNNs) [10] or Generative Adversarial Networks (GANs) [11] prevail in many fields, including image classification, time series forecasting, visual object generation or Reinforcement Learning (RL), being the latter of main interest for this Thesis.

There are several properties of DL models that make them outperform traditional *shallow learning* methods. Among them, DL models can automatically learn hierarchical features from raw data, so that features organized in higher levels of the hierarchy are composed by a combination of simpler lower-level features. As a result of this capability, features with minimal human effort and domain knowledge can be learned and fused together for a manifold of tasks, such as classification, regression or representation learning [12]. Furthermore, DL models comprise a large number of parameters to represent such hierarchical features, which are adjusted (*trained*) as per the task under consideration. In addition, DL approaches can model highly non-linear mappings between their inputs and outputs [13, 14]. Finally, decisions issued by these black-box models can be explained to non-expert users, making these black-box models of practical use in domains where explainability is a must [15].

However, it is not just the continually growing levels of complexity of DL models and the upsurge of new architectures stemming regularly from the literature what confers DL its renowned potential to expand beyond its own milestones. As the knowledge assembled in DL models and their generalization skills are mandatory design goals pursued in this field, the way in which that knowledge can be transformed and adapted to solve new problems fast and efficiently has also grasped significant interest in recent years. In fact, the ability to transfer the knowledge from a DL model designed to solve a certain task to another that deals with a different albeit related task has been widely used to speed up the training time and reduce the quality data requirements in the destination task. More strictly, Transfer Learning (TL [16]) refers to the way in which knowledge learned by a model when addressing a certain task is reused as the starting point for the construction of a model for another task, being of capital relevance in fields like image classification or RL. Following this vein, other approaches concentrate their efforts on the construction of powerful models that can either solve multiple tasks (Multitask Learning or Multitasking [17]) or learn how to adapt efficiently to previously unseen problems (i.e., Meta-learning [18]).

1.2. Optimization Problems in Deep Learning

Under the scope of Artificial Intelligence we can find many evidences of the benefits of combining and fusing different technologies to tackle complex tasks better than using any of them in isolation. DL is not an exception to this statement: the fact that the architectural design, hyper-parameter tuning and training processes of DL can be formulated as optimization problems has motivated a long history between these data-based models and the field of bio-inspired optimization, particularly Evolutionary Computation and Swarm Intelligence. This mixture of technologies, known as Evolutionary Deep Learning (EDL)[1], has been historically studied to optimize some hand-designed parameters of neural networks, such as the number of layers, their dimension and type of neurons, intermediate processing elements (e.g. neural activation functions), and other structural parameters. In fact, the optimization of those parameters can span a large solution space, demanding search heuristics for their efficient exploration. Also, hyper-parameter tuning in DL models can be approached via bio-inspired heuristic wrappers, whereas their training process is essentially the minimization of a task-dependent loss function with respect to all trainable parameters established in the architecture of the network at hand.

It is widely known that a DL model can be seen as a black-box optimizer where some parameters can be manually selected, so that the model behaves in a different way depending on the values chosen for such parameters. In

1 Throughout this manuscript we embrace the term *Evolutionary Deep Learning* to refer to the use of bio-inspired algorithms for solving optimization problems related to DL, no matter if they belong to Evolutionary Computation or to Swarm Intelligence.

fact, almost all parameters that can be tuned in a DL model can be treated as a task to be optimized. Therefore, depending on the parameters to be solved, we can differentiate various optimization problems. To define them properly, we start from the mathematical formulation of a DL model as a composition of N different functions (i.e. *layers*) that maps its input $\mathbf{x}_n \in \boldsymbol{\mathcal{X}}_n$ to an output $\mathbf{y}_n = f_n(\mathbf{x}_n; \boldsymbol{W}_n; T_n, \boldsymbol{\theta}_n) \equiv f_{n,\boldsymbol{W}_n}^{T_n,\boldsymbol{\theta}_n}(\mathbf{x}_n)$, where:

- $T_n \in \mathcal{T}$ denotes the *type of layer*, with $\mathcal{T}$ denoting the set of possible layer types (e.g. Convolutional, Long Short-Term Memory (LSTM)).
- $\boldsymbol{\theta}_n$ is the vector of *structural hyper-parameters* of the layer. The specific parameters in this vector depend on the type T_n of the layer (e.g. $\boldsymbol{\theta}_n$ will specify the sizes of the convolutional filters only if $T_n =$ `convolutional`).
- $\boldsymbol{W}_n$ denotes the *trainable parameters* (weights/filter coefficients and biases) of layer n, whose type and cardinality depend on T_n and $\boldsymbol{\theta}_n$. For instance, if $\mathbf{x}_n$ represents RGB images (3 channels), $T_n =$ `convolutional` and $\boldsymbol{\theta}_n$ establishes that layer n comprises five 3×3 convolutional filters, $\boldsymbol{W}_n$ will comprise $3 \times 3 \times 5 \times 3$ weights and 5 biases, yielding a total of $|\boldsymbol{W}_n| = 140$ trainable parameters.

It is important to highlight that the values of the trainable parameters $\boldsymbol{W}_n$ must be learned by the model to efficiently perform a given task. For the sake of simplicity, in subsequent derivations we will assume that we deal with a supervised learning task over a training dataset $\mathcal{D}_{tr} = \{(\mathbf{x}_1^m, \mathbf{y}_N^m)\}_{m=1}^{M_{tr}}$, with $\mathbf{y}_N^m \in \boldsymbol{\mathcal{Y}}$ denoting the supervised output of input $\mathbf{x}_1 \in \boldsymbol{\mathcal{X}}_1$, and M_{tr} representing the number of training instances. The trainable parameters of a DL model are learned from $\mathcal{D}_{tr}$ by means of a *training* algorithm :

$$\{\boldsymbol{W}_n\}_{n=1}^N = ALG(\mathcal{D}_{tr}, \{T_n, \boldsymbol{\theta}_n\}_{n=1}^N; \boldsymbol{\vartheta}), \tag{1.1}$$

where we refer to $\boldsymbol{\vartheta}$ as the set of *training hyper-parameters* of the training algorithm. In general, the training algorithm is driven by the minimization of a task-dependent loss function $L(\widehat{\mathbf{y}}_N^m, \mathbf{y}_N^m)$ that provides a measure of error between the supervision $\mathbf{y}_N^m$ of input $\mathbf{x}_1^m \in \mathcal{D}$ and the corresponding output of the DL model:

$$\widehat{\mathbf{y}}_N^m = f_{N,\boldsymbol{W}_N}^{T_N,\boldsymbol{\theta}_N} \circ f_{N-1,\boldsymbol{W}_{N-1}}^{T_{N-1},\boldsymbol{\theta}_{N-1}} \circ \ldots \circ f_{1,\boldsymbol{W}_1}^{T_1,\boldsymbol{\theta}_1}(\mathbf{x}_1^m), \tag{1.2}$$

where $\circ$ denotes composition of functions (i.e. $f \circ g(x) = f(g(x))$). Such a loss computed for every training instance can be averaged to yield a numerical estimation of the performance of the DL model when approximating the supervised instances in $\mathcal{D}_{tr}$:

$$L(F; \mathcal{D}_{tr}) = \frac{1}{M_{tr}} \sum_{m=1}^{M} L(F(\mathbf{x}_1^m; \{\boldsymbol{W}_n\}_{n=1}^N; \{T_n\}_{n=1}^N, \{\boldsymbol{\theta}_n\}_{n=1}^N), \mathbf{y}_N^m) \tag{1.3}$$

With this notation in mind, we define the following optimization problems that underlie the construction process of DL models:

Problem 1: Topological Optimization

Given a learning task defined on a training dataset $\mathcal{D}_{tr}$, the topological optimization of a DL model refers to the search for the topology of the DL model that best solves the task at hand, wherein topology involves the discovery of the optimal number of layers N and their types $\{T_n\}_{n=1}^N$. This problem assumes fixed values for $\{\boldsymbol{\theta}_n\}_{n=1}^N$ (e.g. standard values), and relies on a training algorithm $ALG(\mathcal{D}_{tr}, \{T_n, \boldsymbol{\theta}_n\}_{n=1}^N; \boldsymbol{\vartheta})$ to optimize the trainable parameters $\{\boldsymbol{W}_n\}_{n=1}^N$. Mathematically:

$$\min_{N, \{T_n\}_{n=1}^N} L(F; \mathcal{D}_{tr}) \tag{1.4}$$

where the dependence of the aggregate loss function with respect to N and $\{T_n\}_{n=1}^N$ comes through Equation 1.3, and $\{\boldsymbol{W}_n\}_{n=1}^N$ are optimized by means of Equation 1.1.

Topology optimization is rarely conceived in isolation with respect to the rest of variables that define a DL model. Instead, topology is often optimized along with the values of their structural hyper-parameters. However, we define this second problem separately so as to allow for a fine-grained analysis in subsequent elaborations throughout the Thesis:

Problem 2: Structural Hyper-parameter Optimization

Given a learning task defined on a training dataset $\mathcal{D}_{tr}$, and a fixed topology of the DL model (N and $\{T_n\}_{n=1}^N$), the optimization of the structural hyper-parameters of the DL model aims to find the best value of $\boldsymbol{\theta}_n$ (structural hyper-parameters) for each of their compounding layers. Mathematically:

$$\min_{\{\boldsymbol{\theta}_n\}_{n=1}^N} L(F; \mathcal{D}_{tr}) \tag{1.5}$$

where the dependence of the aggregate loss function with respect to variables $\{\boldsymbol{\theta}_n\}_{n=1}^N$ comes through Equation 1.3, and $\{\boldsymbol{W}_n\}_{n=1}^N$ are optimized by means of Equation 1.1.

Finally, the third optimization problem that can be formulated is the training process itself, which aims at finding the values of the parameters $\{\boldsymbol{W}_n\}_{n=1}^N$ that minimizes the loss in Equation 1.3. This is indeed the purpose of Equation 1.1. However, we note at this point that two different formulations of this problem can be made depending on whether variables to be optimized include the set of *training hyper-parameters* $\boldsymbol{\vartheta}$ or *trainable parameters* $\{\boldsymbol{W}_n\}_{n=1}^N$ (i.e. weights):

Problem 3: Training Hyper-parameter Optimization

Given a learning task defined on a training dataset $\mathcal{D}_{tr}$, a fixed topology of the DL model (N and $\{T_n\}_{n=1}^N$), fixed values of their structural hyper-parameters $\boldsymbol{\theta}_n$, and a training algorithm $ALG(\mathcal{D}_{tr}, \{T_n, \boldsymbol{\theta}_n\}_{n=1}^N; \boldsymbol{\vartheta})$, the training hyper-parameter optimization problem of a DL model aims to find the best value of $\boldsymbol{\vartheta}$ (training hyper-parameters) as:

$$\min_{\boldsymbol{\vartheta}} L(F; \mathcal{D}_{tr}) \tag{1.6}$$

where the dependence of the aggregate loss function with respect to $\boldsymbol{\vartheta}$ comes through the application of Equation 1.1 to solve for $\{\boldsymbol{W}_n\}_{n=1}^N$ as per Equation 1.3.

The last problem focuses on the optimization of trainable parameters (i.e. weights). This entails a great challenge through the lenses of optimization, mainly due to the high dimensionality of the search space (mainstream DL models usually account for several millions of trainable parameters). The problem can be summarized as:

Problem 4: Trainable Parameter Optimization

Given a learning task defined on a training dataset $\mathcal{D}_{tr}$, a fixed topology of the DL model (N and $\{T_n\}_{n=1}^N$), and fixed values of their structural hyper-parameters $\boldsymbol{\theta}_n$, the trainable parameter optimization problem of a DL model seeks the best value of $\{\boldsymbol{W}_n\}_{n=1}^N$ (trainable parameters) as:

$$\min_{\{\boldsymbol{W}_n\}_{n=1}^N} L(F; \mathcal{D}_{tr}) \tag{1.7}$$

for which an optimization (training) algorithm (Equation 1.1) must be utilized.

A visual summary of the above problems is sketched in Figure 1.1. The above four optimization problems can represent the majority of contributions that have proposed so far new algorithms to address them efficiently.

There are plenty of reasons to delegate problems 1 to 4 to approximate solvers. However, the two main reasons can be summarized as follows: a) parameters tuned by hand for problems 1, 2 or 3 is prone to errors and tedious trial-and-improve processes, which may be prone to suboptima; and b) optimal parameters for any of the problems may differ between instances of the same problem characterized by different datasets $\mathcal{D}_{tr}$. Furthermore, other limitations appear when using optimization algorithms to overcome these limitations, such as the cost of evaluating the quality of a given candidate solution or the high dimensionality of the search space, especially in – yet not limited to – Problem 4.

Considering these limitations, Problem 4 can be considered as an exceptionally complex optimization challenge. The use of optimization algorithms

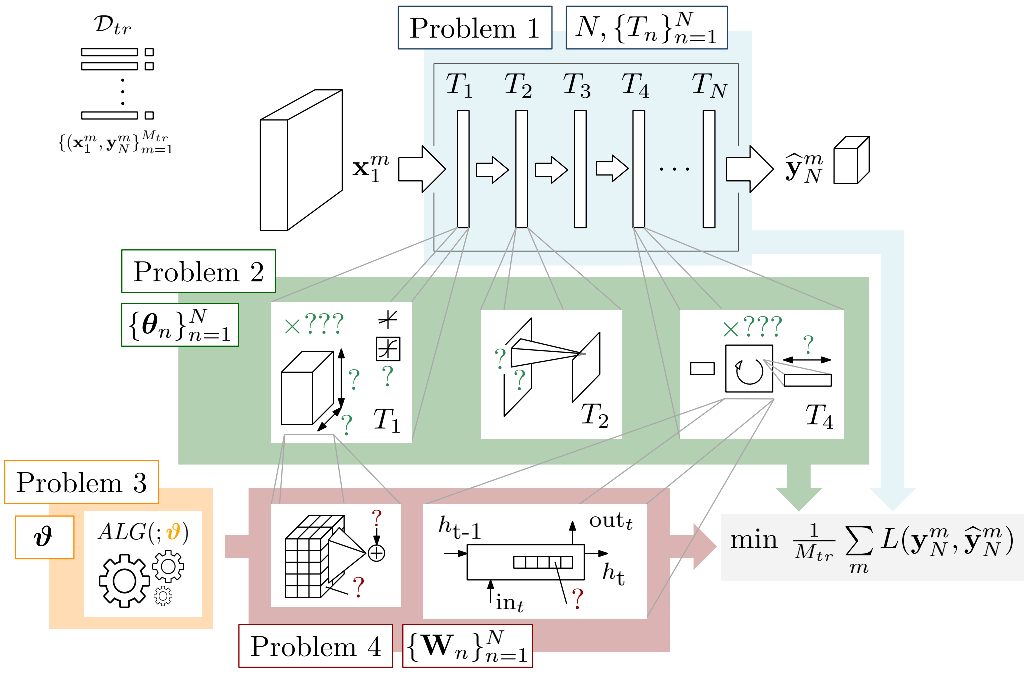
$\mathcal{D}_{tr}$
$\{(\mathbf{x}_1^m, \mathbf{y}_N^m\}_{m=1}^{M_{tr}}$
Problem 1
$N, \{T_n\}_{n=1}^N$
T_1 T_2 T_3 T_4 T_N
$\mathbf{x}_1^m$
$\widehat{\mathbf{y}}_N^m$
Problem 2
$\{\boldsymbol{\theta}_n\}_{n=1}^N$
×???
?
T_1
T_2
T_4
Problem 3
$\boldsymbol{\vartheta}$
$ALG(;\boldsymbol{\vartheta})$
h_{t-1}
out$_t$
h_t
in$_t$
Problem 4
$\{\mathbf{W}_n\}_{n=1}^N$
$\min \frac{1}{M_{tr}} \sum_m L(\mathbf{y}_N^m, \widehat{\mathbf{y}}_N^m)$

Figure 1.1: Optimization problems in Deep Learning for a generic model comprising, among others, a convolutional layer, a max-pooling layer and a recurrent layer.

to solve them effectively is not conceived to alleviate the potential issues of human-driven network architecture and/or hyperparameters configuration steps, but rather to replace the gradient backpropagation algorithm that is mostly used to tackle Problem 4. Solving Problem 4 by relying on back-propagated gradients is also known to undergo severe limitations, such as its vanishing/exploding behavior in networks of moderate to high depth. In contrast, evolutionary algorithms (and in general, mmeta-heuristic optimization algorithms) have demonstrated to effectively balancing exploration and exploitation in complex search spaces, do not rely on any gradient-based information, and can be flexibly adapted to search not only for optimality, but also for other objectives (e.g. complexity of the network or diversity among several solutions to the problem under consideration). Therefore, evolutionary algorithms has been postulated as a potential replacement for gradient backpropagation, being explored mostly for networks of small-to-medium size.

Up to this point, DL problems usually tackled under the scope of EDL have been introduced. The design of good solvers capable of facing them is not trivial, and there are plenty of mmeta-heuristic optimization algorithms that can yield similar outcomes for any instance of the above problems. For this reason, in what follows Section 1.3 delves into the classification of mmeta-heuristic algorithms based several different criteria.

1.3. Bio-inspired Optimization: Evolutionary Computation and Swarm Intelligence

The need for search algorithms capable of efficiently dealing with the optimization problems arising from DL has stimulated an upsurge of literature proposing different solvers for this purpose. Here, a brief overview of the optimization research area is provided, with a focus on meta-heuristic algorithms that are inspired by biological sources of inspiration. For a more detailed overview of developments and prospects in this research area we refer to recent comprehensive reviews on this topic available in [19, 20].

As shown in the taxonomy shown in Figure 1.2, optimization methods can be first grouped in three categories: exact methods, heuristics and meta-heuristics. Exact methods are those that always solve a problem to its optimality, either by exploring the entire space of solutions or by taking advantage of specific characteristics of the problem at hand (e.g. linearity, convexity). On the other hand, a heuristic search algorithm addresses a given optimization problem by resorting to knowledge related to the domain where the problem is formulated. By exploiting this domain-specific information in its search algorithm, a heuristic explores the space of feasible solutions efficiently, intensifying the search around the most promising areas as per the objective(s) under consideration. Finally, the third category corresponds to meta-heuristic algorithms (also referred to as *mmeta-heuristics*), which are central to the research hypothesis of this Thesis.

Briefly explained, a meta-heuristic optimization algorithm solves a problem using only general information and knowledge common to a wide variety of problems with similar characteristics [21]. Meta-heuristic algorithms explore the solution space by progressively learning how candidate solutions should be modified towards optimality, with the aim of reaching increasingly promising results disregarding the characteristics of the problem being tackled. Given their self-learning nature and their abstraction from the problem itself, meta-heuristic approaches are well-suited to deal with real-world problems featuring complex search spaces and even non-analytically defined objectives/constraints. This is in fact the reason why meta-heuristics have taken a prominent role when addressing the optimization problems underneath DL.

Deeper into the taxonomy of Figure 1.2, meta-heuristics can be further divided into different groups depending on several criteria. To begin with, we can distinguish between 1) single-point (also referred to as trajectory-based) meta-heuristic methods, which rely on the progressive improvement of a single solution to the problem by exploring its neighborhood under a set of movement operators as in Tabu Search (TS [22]) or Simulated Annealing (SA [23]); and 2) population-based techniques, which maintain a set of possible solutions of the problem that interact with each other towards producing new solutions of increased quality (e.g. Genetic Algorithm (GA [24, 25]), Ant Colony Optimization (ACO [26]) or Particle Swarm Optimization (PSO [27])). This last category can be broken down in 2.1) multi-population techniques,

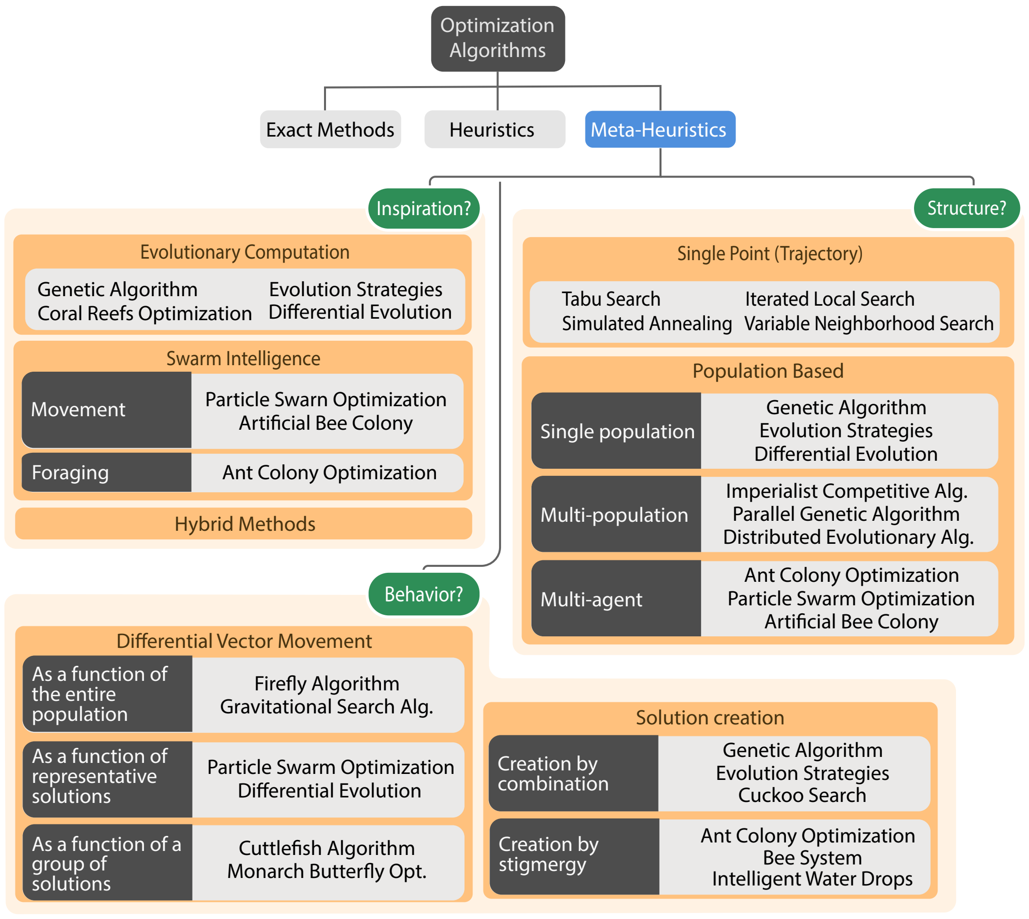

Figure 1.2: Taxonomy of optimization algorithms, with a focus on meta-heuristic optimization algorithms as per the different criteria under which they can be classified. Some examples of algorithms are also given.

where the population is divided into different subpopulations that evolve separately, exchanging information periodically (for instance, the Imperialist Competitive Algorithm [28]); 2.2) multi-agent methods, whose population is composed by multiple diverse agents with different roles that interact with each other towards reaching increasingly better solutions (e.g. Artificial Bee Colony (ABC [29])); and 2.3) single-population approaches, such as the aforementioned GA. At the same time, meta-heuristics can also be divided as per its search behavior, yielding A) differential vector movement based methods, which rely on the computation of a differential vector to move from a reference solution towards a new candidate; and B) solution creation based methods, which generate new solutions to explore the search space instead of evolving existing ones incrementally. Other criteria can be adopted for organizing the enormous corpus of literature related to optimization meta-heuristics, such as the support of the optimization variables of problems that can be tackled by the meta-heuristic at hand (discrete/continuous/mixed), the scope of the search (global/local), or the stochastic/deterministic nature of their search operators, among other criteria.

Among these criteria, the inspiration underneath the search algorithm itself has sprung a vast area of research widely known as bio-inspired optimization [30]. Over the last decades, a manifold of behavioral patterns observed in biological systems have been emulated to yield intelligent algorithms capable of mimicking the learning and adaptation capabilities of such biological systems to address complex computational problems. Therefore, a bio-inspired meta-heuristic algorithm can be categorized as such if its main search strategy gets partially or fully inspired by biological phenomena, such as the evolution of species, the echolocation of bats or the foraging behavior of ant colonies. A plethora of inspiring metaphors can be found nowadays in contributions dealing with new bio-inspired optimization algorithms, not without an ongoing controversy on the value of the metaphor itself for the novelty and scientific soundness of the reported methods [19].

Leaving such disputes aside, a research trend that has so far endured over the years is the hybridization of bio-inspired algorithms with problem-specific local search methods. The main reason behind this practice is to exploit the advantages of bio-inspired solvers, and to overcome their disadvantages when dealing with problems for which ad-hoc heuristics can be developed and inserted into the overall search process. Arguably, Memetic Algorithms [31] capitalize on this principle, with many application domains having so far harnessed this synergy between global and local search algorithms [32].

1.4. Motivation and Hypothesis

In the last years the scientific community has reported outstanding results on the application of mmeta-heuristics for the hyperparameter tuning, structural hyperparameter tuning, and architectural design problems introduced in Section 1.2. Nonetheless, when it comes to trainable parameter optimization (Problem 4), the literature falls short in providing evidences that mmeta-heuristics are a viable choice for the purpose. Experimental benchmarks are far from considering realistic network sizes and levels of complexity, placing the starting point of the benchmarks in favor for the adoption of meta-heuristic optimization algorithms. Furthermore, comparisons appear to be frequently biased by only focusing on statistics about the quality of the produced solutions, without examining the implications of their use in terms of computational effort. As a result, even if the interest in training neural networks using mmeta-heuristics (namely, EDL) has raised prominently in the last years, there remains an halo of doubt around the potential of these solvers to replace well-established gradient-based techniques for neural training.

There emerges the first motivational factor for this Thesis: to perform a thorough inspection and critical analysis of the literature related to EDL, and to experimentally assess the potential of this research area to undertake the problems stated previously. The lack of prior studies covering this niche in the community stimulates a first research effort to perform this analysis, serving in turn as an informative stepping stone towards the ultimate goal of this Thesis: to showcase new trends in bio-inspired optimization that can make a difference in modeling tasks involving training several neural networks at once.

In this regard, a new optimization paradigm forged as Evolutionary Transfer Optimization (ETO) has brought a fresh breeze to the community, analyzing how the solutions evolved by evolutionary operators for a given problem can be exploited to boost the convergence of the mmeta-heuristic search processes of other different problems(s), transferring knowledge about the search space and/or solutions either sequentially or simultaneously. Even though classification and regression tasks formulated as optimization problems may not profit significantly from the use of ETO algorithms, this Thesis exposes some scenarios that could lend themselves to a more beneficial and synergistic mix of technologies between DL and ETO. This is the case of some RL setups comprising different environments/tasks, including multitask RL (i.e. knowledge transfer between tasks/models for the agents to solve them in a better/faster fashion) or meta RL (i.e. knowledge transfer between models facing different tasks through time). The hypothesis of the Thesis in this matter is that the transfer of knowledge realized by ETO can be helpful to expedite the convergence of the training processes of RL agents when they tackle environment/tasks that are related to each other (multitask RL), or to consolidate primal knowledge between agents, suited to generalize better to unseen tasks over time (meta RL).

The Thesis elaborates on these motivations and hypothesis, unleashing therefrom a number of objectives and pursued milestones that are enumerated and described in the next section.

1.5. Objectives of the Thesis

In this section the main objectives of the Thesis are described. As anticipated previously, the main target is to explore and study new synergies between Evolutionary Computation and DL. This ultimate goal can be broken down in a number of specific objectives, detailed as follows:

- *Objective 1* - **Literature Review and Analysis of EDL and ETO**

  This specific goal is covered in Chapter 2 and Chapter 3, where exhaustive studies of the state of the art in the field of EDL and ETO is carried out. Together, these two first chapters comprise the second constituent part of the Thesis (Part II), in correspondence to the first introductory part of the Thesis (Part I, which spans the present chapter):

  To begin with, the literature review in Chapter 2 is conducted by examining different trends in EDL followed to solve the problems introduced in Section 1.2. Based on the conclusions drawn from this analysis and the weaknesses spotted in Section 2.3, two empirical experiments are carried out to shed light over two questions derived from the analysis: i) How does bio-inspired solvers perform when used to optimize the architecture and hyperparameters of a multi-layered neural networks?; and ii) Is a bio-inspired algorithm specifically designed for Large Scale Optimization (SHADE with Iterative Local Search (SHADE-ILS [33])) enough to train neural networks formed by thousands of parameters? If not, is there any scheduled or layer-wise way to decompose the problem and make it more approachable by using?

On the other hand, the review in Chapter 3 gravitates on on different methodologies under the ETO paradigm, stressing on their differences regarding how knowledge is shared among the optimization problems at hand. From the study of the literature, insights and critical aspects of this recent optimization branch are offered. As a result, the reader will get familiar with the mathematical notations and concepts underneath multi-factorial algorithms that will be thereafter used in the contributions of the Thesis presented in Part III.

- *Objective 2* - **Expose the potential of ETO for knowledge transfer in multitask and meta RL [Part III]**

  Nourishing from the findings learned from the literature reviews in the preceding chapters, Part III shows that ETO can help certain modeling tasks approached via DL when they involve transferring knowledge between different modeling instances. This third part of the Thesis targets this goal by addressing two different scenarios where this assumption holds:

  - *Objective 2.1* - **Evolutionary Multitasking for RL** [Chapter 4]

    In this first scenario, a Multifactorial Optimization (Multi Factorial Optimization (MFO)) algorithm coined as Multi Factorial Evolutionary Algorithm (MFEA [34]) is used to simultaneously train multiple RL tasks formulated as optimization algorithms. By means of an adapted version of MFEA and a tailored design of the unified search space, MFEA's internal operators are used to leverage inter-task relationships by transferring knowledge between such related tasks, mimicking the well-known and widely used TL approach. This contribution proves that MFEA can train efficiently multiple RL tasks, taking advantage of the exploratory skills of mmeta-heuristics, and the capability of MFO to exchange knowledge between related problems during the search.

  - *Objective 2.2* - **Evolutionary Meta-learning for RL** [Chapter 5]

    A step beyond the multitask setup tackled in the previous chapter is the one described in Chapter 5. Specifically, the knowledge evolved by ETO is further manipulated and processed, to yield a set of models capable of better solving unseen tasks. This is accomplished by a methodology that permits to automatically define which tasks can be merged together and the *essential knowledge concepts* prevalent in the evolved models. To this end, models evolved by ETO are clustered using a measure of distance between trained neural networks. Finally, newly appearing tasks are tested over the representative models of such concepts assuming zero- (no feedback from unseen tasks) and few-shot learning (restricted chance to update the models based on the feedback from unseen tasks).

1.6. Methodology

This section describes the research methodology followed in the development of the research lines integrated in this Thesis. The methodology lies on the roots of the scientific method, slightly adapted to the specific requirements of the Thesis. First, the conclusions conducted from an exhaustive analysis of the literature are used to support or adapt initial hypotheses. Once evidences are enough and robust to support the hypotheses, the development of the technical contributions of the Thesis begins, presenting the results to the community in high-impact conferences or journals. Replicability and transparency are two principles of the Thesis: code to reproduce the experiments herein reported are publicly available in renowned software repositories, easily accessible for anyone.

1.7. Structure and Reading this Thesis

Figure 1.3 summarizes the structure and reading flow of the Thesis. Part I has introduced the concepts of DL, bio-inspired mmeta-heuristic optimization, and the synergies between these two paradigms. Additionally, the motivation and objectives to pursue through the rest of the Thesis have been posed. The first objective of the Thesis is approached in Part II, where an exhaustive review of the state of the art and current trends on EDL (Chapter 2) and ETO (Chapter 3) is performed. Then, objectives 2.1 and 2.2 are addressed in Part III. Chapter 4 focuses on adapting MFEA to efficiently solve multiple RL tasks simultaneously. Then, Chapter 5 builds upon its predecessor by facing even more complex scenarios, where the objective is to perform meta-learning over a set of RL tasks. Finally, Part IV summarizes the contributions of the Thesis, details the publications derived from this work, and outlines future research directions.

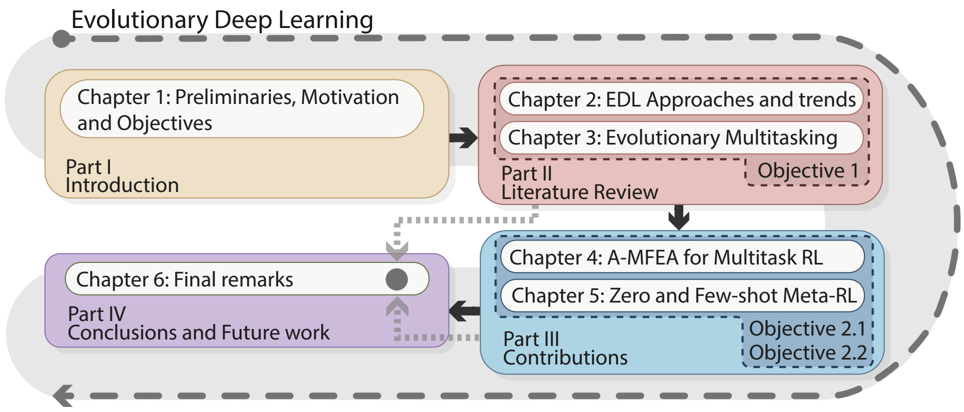
Evolutionary Deep Learning
Chapter 1: Preliminaries, Motivation and Objectives
Part I
Introduction
Chapter 2: EDL Approaches and trends
Chapter 3: Evolutionary Multitasking
Objective 1
Part II
Literature Review
Chapter 4: A-MFEA for Multitask RL
Chapter 5: Zero and Few-shot Meta-RL
Objective 2.1
Objective 2.2
Part III
Contributions
Chapter 6: Final remarks
Part IV
Conclusions and Future work

Figure 1.3: Diagram showing the structure and reading flow of the Thesis.

Part II

LITERATURE REVIEW

2 EVOLUTIONARY DEEP LEARNING: APPROACHES AND CURRENT TRENDS

"Be curious on your journey!"
- The Outer Wilds

2.1. Introduction

The purpose of this chapter is to perform a thorough assessment of the potential of meta-heuristic algorithms for DL (i.e. Evolutionary Deep Learning (EDL)), supporting an informed understanding of the current state of the art of this research area. It is supported by an exhaustive critical examination of the recent literature falling in this intersection, and a profound reflection, informed with empirical results, on the lights and shadows of this research avenue.

To this end, the contributions reported in this area over the years are categorized, comprehensively reviewed and critically examined based on three axes: a) *optimization and taxonomy*, which comprises a historical perspective on this fusion of technologies, connecting clearly with the optimization problems in DL defined in Chapter 1, and a taxonomy associated to an in-depth analysis of the literature; b) *critical analysis*, informed by two case studies, which altogether elicit a number of lessons learned and recommendations for good practices, and c) *challenges* that motivate new directions of research for the near future like the ones in Part III of the Thesis. These three axes of this study aim to provide a clear response to four important questions related to EDL, which are represented in Figure 2.1.

Throughout the rest of this chapter, Section 2.2 briefly overviews the historical connection between DL and bio-inspired optimization. Next, Section 2.3 presents the taxonomy and an analysis of the literature falling on each of its categories. Section 2.4 exposes methodological caveats resulting from the critical literature study. Section 2.5 present the two designed cases of study, and discuss the results obtained therefrom. Section 2.6 enumerates learned lessons and prescribes good practices to be followed by prospective studies. Section 2.7 outlines several challenges and research directions that should drive future research efforts of the interested audience. Finally, Section 2.8 points out the final outline and conclusions of this chapter.

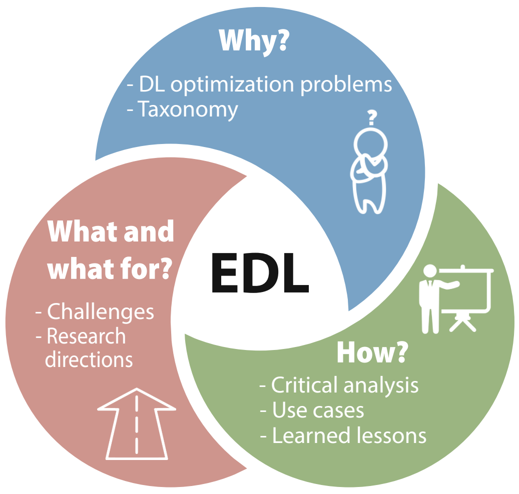

Figure 2.1: Diagram depicting the three axes and four fundamental questions on Evolutionary Deep Learning tackled in this chapter, along with the specific aspects that contributes to each question.

2.2. Evolutionary Deep Learning

Despite its relative youth, the current momentum of the synergy between DL and bio-inspired optimization is founded on a set of historical milestones that suggested the scientific community to combine these two branches of Artificial Intelligence. Herein, the background that led into the literature mainstream that motivates the current study is briefly revisited. Figure 2.2 summarizes graphically such milestones, arranging them in a timeline along with the number of related publications reported in the last few years.

Although timid attempts at NeuroEvolution (NE) with bio-inspired solvers had been reported in the late 90s [35], it was not until 2002 when Stanley and Miikkulainen settled a major breakthrough in the research community with their seminal work "Evolving neural networks through augmenting topologies". The NeuroEvolution of Augmenting Topologies (NEAT [36]) approach proposed in this work allowed connection and layer types of an Artificial Neural Network (ANN) architecture to be optimized by means of a meta-heuristic algorithm towards a progressively better precision of the evolved model for a given task. NEAT embraces the main workflow of population-based meta-heuristics, particularly genetic algorithms: a population of encoded candidates is generated representing several network architectures, from which new candidate architectures are produced and evaluated on a given task (in the original work, a RL task). After all candidates in the population have been evaluated, mutation and crossover operators are applied, generating a new population by means of combining network architectures, generating new layers or varying their hyper-parameters. This iterative search process is stopped when a stopping criterion set beforehand is met. As just stated, this algorithm was mainly proposed to solve reinforcement learning tasks, keeping in mind the profit of applying meta-heuristics to such environments, such as getting interesting behaviours and not falling in local optima in expense of precision.

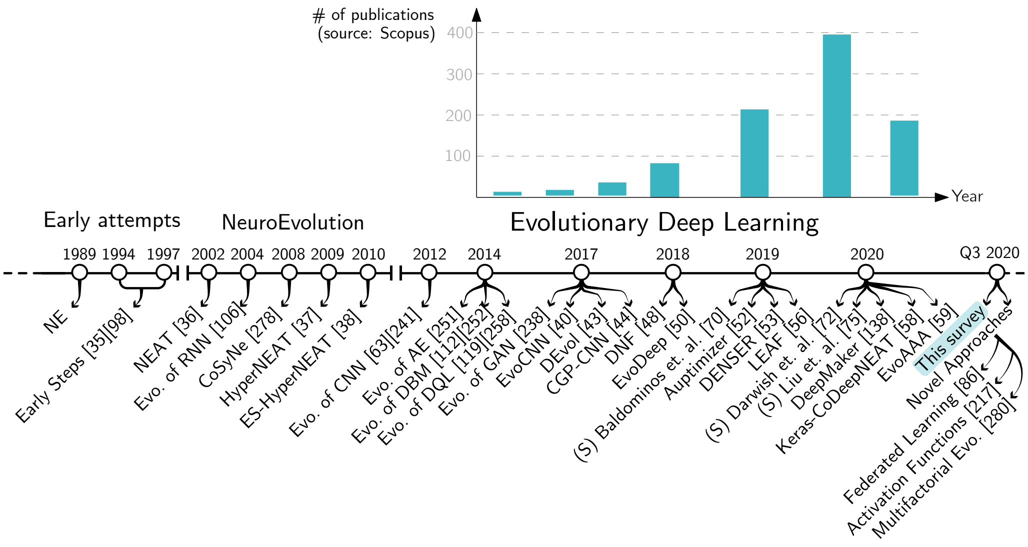

Figure 2.2: Timeline with main milestones in the history of Evolutionary Machine Learning, and a bar diagram showing the number of publications reported in this research area during the period 2013-June 2020. Data retrieved from Scopus by submitting the query (`EVOLUTIONARY` OR `SWARM INTELLIGENCE`) AND `DEEP LEARNING`. (S) stands for *Survey*.

Shortly after its first publication, NEAT's unprecedented results spurred a flurry of new extensions and variants, not only in terms of new tasks and applications, but also in what refers to its core algorithmic components. Regarding the latter, the acknowledged importance of using good network encoding strategies soon became a major research goal in this literature strand, given the huge search space spawned by the evolution of architecture and weights of ANN. In its original version, NEAT encoded candidates using a direct encoding strategy, i.e. networks' hidden units, connections and parameters (phenotype) were *directly* represented as an array representing each point of the network (genotype). However, in 2009 Stanley et al. proposed the so-called Hypercube-based NEAT, or HyperNEAT [37], which relies on a generative encoding strategy to evolve large-scale neural networks using geometric regularities of the task domain and compositional pattern producing networks (namely, an ANN variant comprising multiple potentially heterogeneous activation functions that can be evolved via genetic algorithms). Besides the optimization of the activation function of each neuron in the network, HyperNEAT also proposed the use of an indirect encoding to represent the networks to be evolved, inheriting other concepts from preceding NEAT versions such as speciation and historical marking, which were also adopted and extended in ES-HyperNEAT [38]. Years later, a new NEAT approach was developed for CNNs, which was coined as CoDeepNeat [39]. Following the NEAT design principles, CoDeepNeat was proved to excel at evolving layers, parameters, topology and hyper-parameters of CNNs. To this end, it uses an indirect encoding approach so that the network information can be encoded as rules or processes for creating individuals, ultimately yielding a reduced representation of the search space that can be explored more efficiently by the bio-inspired algorithm in use.

Since its first appearance, NE approaches (with NEAT at their forefront) have been applied to multitude of tasks and problems. A major fraction of them relate to RL, such as car controllers [60] or first-person agents control [61]. Interestingly, years after these applications were reported, the community shifted its research focus towards bio-inspired algorithms as an efficient replacement to train Deep Reinforcement Learning methods [46], showing up competitive results. Bio-inspired optimization was also applied to other DL tasks at the time, particularly for CNNs [62], and showcased in a diversity of applications including the classification of epileptic EEGs [63], time series forecasting [64] or scheduling deicing tasks in airports [65].

As a result of these findings, a growing literature corpus started to explore the potential of neuro-evolution strategies for the topology and structural hyper-parameter optimization of different DL models, including Auto-Encoder (AE), Deep Boltzmann Machine (DBM), and GANs. Naturally, the research community soon flowed into a further use of bio-inspired algorithms: the optimization of the trainable parameters of DL models [66]. An early approach was explored in [67], where a genetic algorithm is used for architecture optimization, and differential evolution for weight optimization. The main scientific interest of this work and those published thereafter is to assess whether meta-heuristics can avoid the convergence limitations of gradient descent methods when facing strongly non-convex search spaces as those characterizing the problem of training complex DL architectures. Nonetheless, most agreed on the dilated computation time and enormous computational resources needed by meta-heuristic solvers, which outweigh in practice the eventual convergence gains reported in experimental studies. As a result, gradient back-propagation approaches have dominated as the DL training solvers of choice over the years. Recently, this tendency is again emerging, stimulated by advances in highly parallel computing paradigms (including GPU and distributed asynchronous computation), prospecting new ways to train DL models with bio-inspired algorithms appearing frequently as promising alternatives.

Although it is still an incipient research field, some huge companies like Google, Facebook or Uber have glimpsed the power of this hybridization, and have started investing massively in software frameworks towards optimizing their DL models with meta-heuristics. The research community has also joined this momentum by open-sourcing software packages for this same purpose. EvoDeep [50], AutoKeras [68] or Google Cloud AutoML are noteworthy platforms used for autonomously optimizing DL models, which are collected in Table 2.1 together with other alternatives from the literature.

The complexity of these problems, given by the cardinality of their search spaces and/or the large number of variables to be optimized, has stimulated an ever-growing corpus of literature that lasts to date. New terms such as Neural Architecture Search (NAS) have been forged to collectively refer to all techniques aimed at automating the design of neural networks. For this

Table 2.1: Software frameworks to evolve Deep Learning models using meta-heuristics and other search strategies.

Ref.	Name	Year	DL models	Optimization domains	Optimization algorithm	Programming Language
[40]	EvoCNN	2017	CNN	Topology, structural hyper-parameters, weight initialization	GA	Python
[41]	ATM	2017	DBN (and shallow)	Optimization of hyperparameters and model selection	Bayesian Opt.	Python
[42]	MetaQNN	2017	CNN	Topology, structural hyper-parameters	RL	Python, Caffe
[43]	DEvol	2017	CNN	Topology, structural hyper-parameters	GA	Python, Keras
[44]	CGP-CNN	2017	CNN	Topology, structural hyper-parameters	CGP	Python, Chainer
[45]	AdaNet	2017	Architectures represented as a DAG	Topology, structural hyper-parameters, training hyper-parameters, trainable parameters	AdaNet	Python, Tensor-Flow
[46, 47]	AI Labs NE Algorithms	2017-2018	CNN	Trainable parameters (weights and biases)	GA + NS	Python, Tensor-Flow
[48]	Auto-Pytorch	2018	CNN + shallow	Topology, structural hyper-parameters, training hyper-parameters	Random-forest-based Bayesian optimization	Python, Pytorch
[49]	DNF	2018	CNN	Trainable parameters (weights and biases)	PSO, HS and DE	Python, Tensor-Flow
[50]	EvoDeep	2018	CNN	Topology, structural hyper-parameters, training hyper-parameters	EA	Python, Keras, Tensorflow
[51]	ENAS	2018	CNN, RNN	Topology	Policy gradient-based subgraph selection	Python, Tensorflow
[52]	Auptimizer	2019	Interface for multiple architectures	Hyperparameter optimization	9 Methods (from grid search to NAS)	Python
[53]	DENSER	2019	CNN	Topology, structural hyper-parameters, training hyper-parameters	GA, DSGE	Python
[54]	Google Cloud AutoML	2019	CNN, RNN	Topology, structural hyper-parameters, training hyper-parameters	TL + NAS	Propri-etary
[55]	AutoKeras	2019	CNN, RNN	Topology	Hyperband, Random, Greedy	Python, Keras
[56]	LEAF	2019	CNN, RNN	Topology, structural hyper-parameters, training hyper-parameters	CoDeepNEAT	Python, Pytorch
[57]	Ludwig	2019	CNN, LSTM, RNN, Fully-Connected	Structural and training hyper-parameters	Tree of Parzen Estimators	Python, Tensor-Flow
[58]	Keras-CoDeepNEAT	2020	CNN, RNN	Topology, structural hyper-parameters, training hyper-parameters	CoDeepNEAT	Python, Keras
[59]	EvoAAA	2020	AE	Topology	GA, ES, DE	R

Table 2.2: Recent overviews on Evolutionary Deep Learning and related topics.

Survey	Period	# reviewed works	Taxonomy	Coverage (DNN models/tasks)	Empirical study	Lessons learned and challenges
[69]	1987-2016	~ 20	Yes (optimization domain of the neural network*)	CNN, RNN, RL, DBN	No	Relevance of data quality. Evolutionary techniques good at exploration and exploitation but no single method for all optimization tasks.
[70]	2014-2018	~ 50	No (temporal analysis)	CNN, RL	No	High computational resources are required. Special emphasis on ensembles, transfer learning, multiobjective and modular evolutionary approaches.
[71]	2011-2018	~ 20	Yes (NN-based or GP-based and optimization problem: architecture, training, multi-objective)	CNN	No	Lack of mathematical foundations, computational costs, scalability, poor generalization ability of the evolved model, lack of interpretability
[72]	2011-2019	~ 90	Yes (Evolutionary/Swarm Intelligence and DL model)	CNN, DBN, RNN, AE	No	Lack of rigurosity by the community. Costs of implementation, run time and overfitting.
[73]	2012-2019	~ 20	Yes (meta-heuristic and DL Architecture)	CNN, DBM, DBN	No	Time, more efforts on enhancing convergence speed and complexity (meta optimization)
[74]	2002-2020	~ 30	Yes(Data preparation, Feature engineering, Model generation and Model evaluation). Completely centered on NAS approaches	CNN, GAN, LSTM	No	NAS applied to more fields. Reproducibility of the experiments, Interpretability and Robustness as main objectives. Seek of a Complete AutoML Pipeline.
[75]	1994-2020	~ 130	Yes (NAS encoding strategy, classification criteria, operators and selection strategy that speed up the evolution)	CNN, DBN, RNN, AE	No	EC based NAS vs Random Search based. Effectiveness of crossover and mutation over just mutating. Acceleration in evaluation as a challenge. A common platform for the comparison needs to be built in the future.
This Thesis	1994-2020	215	Yes (DL model/task and optimization problem)	CNN, AE, DBM, DBN, RNN, GAN, RL	Yes	See Sections 2.6, 2.7 and 2.8

end, Evolutionary Computation and Swarm Intelligence meta-heuristics have been identified among the most interesting search strategies to be developed in years to come, along with reinforcement learning, Monte Carlo Tree Search and other assorted methods [76]. Advances in the use of these meta-heuristics for problems related to DL have been reviewed in a number of surveys on this topic, listed in Table 2.2. However, the critical inspection of the achievements in this area reported over the years reveals poor methodological practices, unsolved technical caveats and research challenges that deserve a detailed analysis of where the community stand in this effervescent area. We now delve into this matter in depth.

2.3. Taxonomy

In light of the past history between bio-inspired optimization and DL, a need arises for properly organizing contributions to date in a taxonomy that covers which problems are addressed, which DL models are involved, and which bio-inspired algorithms are in use. In this section this analysis is performed, centering the discussion around a taxonomy that sorts the literature according to the three aforementioned criteria. The main purpose of this taxonomy and the literature analysis made over each of its categories (Subsections 2.3.1, 2.3.2 and 2.3.3) is to highlight those areas where the community has so far placed most research efforts. This literature analysis settles a firm stepping stone towards a critical discussion of poor methodological practices and points of improvement observed in related contributions to date: the *shadows* in which this field is held nowadays. Such a discussion will be held in Section 2.4.

As has been stated in Section Section 1.2, four main optimization tasks are distinguished: topological optimization (Problem 1), structural hyper-parameter optimization (Problem 2), training hyper-parameter optimization (Problem 3) and trainable parameter optimization (Problem 4). The taxonomy gathers Problem 2 and Problem 3 under the general hyper-parameter tuning category, discriminating between them in a lower level of the taxonomy. The main reason for this special arrangement of the taxonomy is to highlight that as per the reviewed literature, there is little explicit distinction between structural hyper-parameter and training hyper-parameter optimization in related contributions. The thorough examination of this corpus has discriminated interesting research opportunities in the extrapolation of studies and frameworks, from training to structural hyper-parameter tuning and vice versa. When it comes to Problem 4, a distinction is made between i) bio-inspired algorithms that do not incorporate any problem-specific knowledge in their design; and ii) bio-inspired solvers that are hybridized with local search solvers or combined with gradient back-propagation techniques.

Figure 2.3 depicts graphically the taxonomy considered for the literature analysis. The first level considers the type of optimization problem (topology, hyper-parameter and trainable parameter optimization), followed by contributions sorted as per the DL model and kind of bio-inspired solver (Section 1.3) under choice. In what follows representative contributions classified within each of these categories are discussed.

2.3.1. Topology Optimization

It is widely acknowledged that the topology or architecture of DL models have a direct impact on their performance. For this reason researchers have traditionally striven to develop automated methods for generating topologically small yet well-performing network architectures. In this context, the set of algorithms gathered under the NE label aim at progressively augmenting the complexity of neural network topologies to attain increasingly

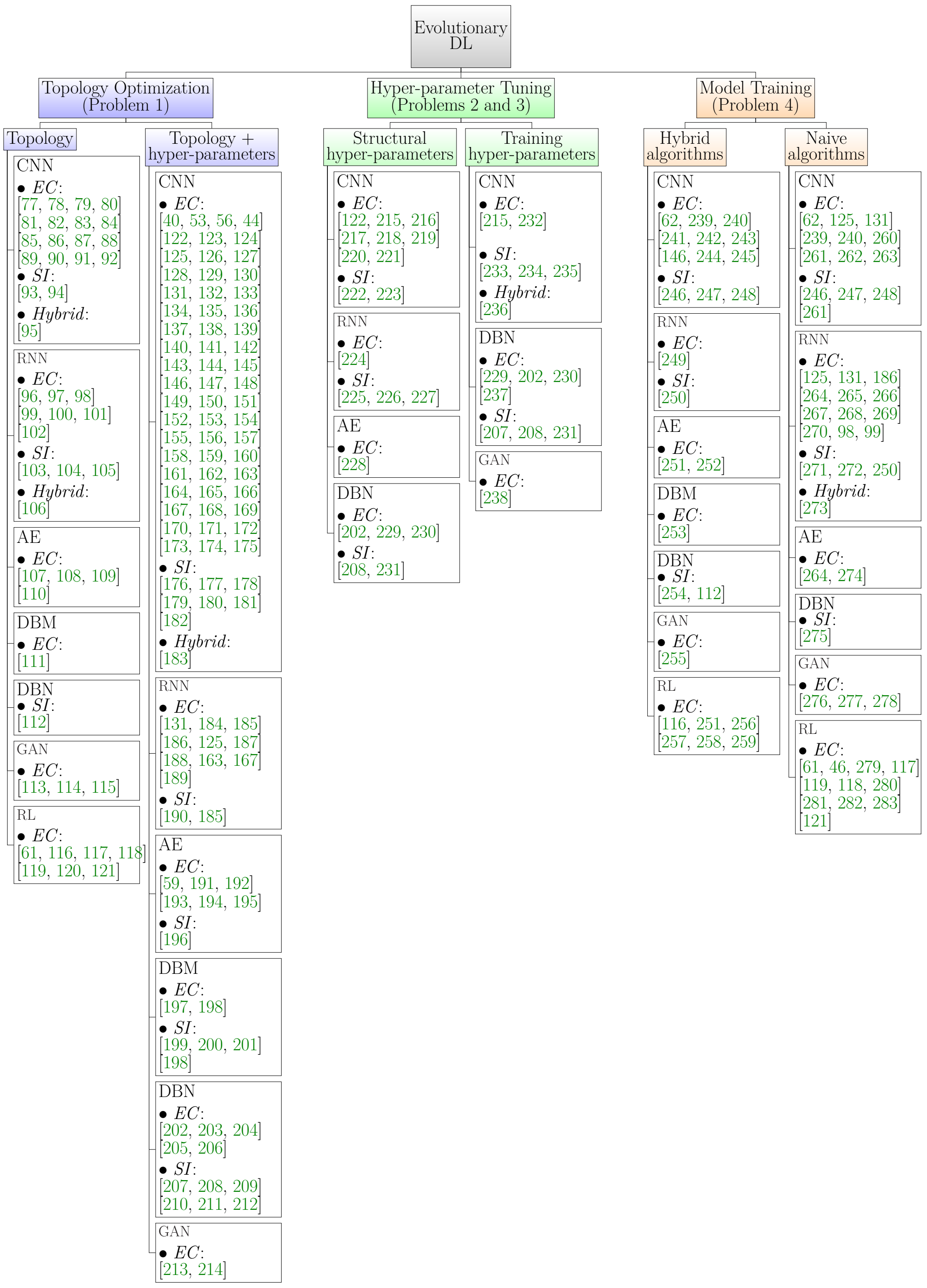
Evolutionary DL
Topology Optimization (Problem 1)
Hyper-parameter Tuning (Problems 2 and 3)
Model Training (Problem 4)
Topology
CNN • EC: [77, 78, 79, 80] [81, 82, 83, 84] [85, 86, 87, 88] [89, 90, 91, 92] • SI: [93, 94] • Hybrid: [95]
RNN • EC: [96, 97, 98] [99, 100, 101] [102] • SI: [103, 104, 105] • Hybrid: [106]
AE • EC: [107, 108, 109] [110]
DBM • EC: [111]
DBN • SI: [112]
GAN • EC: [113, 114, 115]
RL • EC: [61, 116, 117, 118] [119, 120, 121]
Topology + hyper-parameters
CNN • EC: [40, 53, 56, 44] [122, 123, 124] [125, 126, 127] [128, 129, 130] [131, 132, 133] [134, 135, 136] [137, 138, 139] [140, 141, 142] [143, 144, 145] [146, 147, 148] [149, 150, 151] [152, 153, 154] [155, 156, 157] [158, 159, 160] [161, 162, 163] [164, 165, 166] [167, 168, 169] [170, 171, 172] [173, 174, 175] • SI: [176, 177, 178] [179, 180, 181] [182] • Hybrid: [183]
RNN • EC: [131, 184, 185] [186, 125, 187] [188, 163, 167] [189] • SI: [190, 185]
AE • EC: [59, 191, 192] [193, 194, 195] • SI: [196]
DBM • EC: [197, 198] • SI: [199, 200, 201] [198]
DBN • EC: [202, 203, 204] [205, 206] • SI: [207, 208, 209] [210, 211, 212]
GAN • EC: [213, 214]
Structural hyper-parameters
CNN • EC: [122, 215, 216] [217, 218, 219] [220, 221] • SI: [222, 223]
RNN • EC: [224] • SI: [225, 226, 227]
AE • EC: [228]
DBN • EC: [202, 229, 230] • SI: [208, 231]
Training hyper-parameters
CNN • EC: [215, 232] • SI: [233, 234, 235] • Hybrid: [236]
DBN • EC: [229, 202, 230] [237] • SI: [207, 208, 231]
GAN • EC: [238]
Hybrid algorithms
CNN • EC: [62, 239, 240] [241, 242, 243] [146, 244, 245] • SI: [246, 247, 248]
RNN • EC: [249] • SI: [250]
AE • EC: [251, 252]
DBM • EC: [253]
DBN • SI: [254, 112]
GAN • EC: [255]
RL • EC: [116, 251, 256] [257, 258, 259]
Naive algorithms
CNN • EC: [62, 125, 131] [239, 240, 260] [261, 262, 263] • SI: [246, 247, 248] [261]
RNN • EC: [125, 131, 186] [264, 265, 266] [267, 268, 269] [270, 98, 99] • SI: [271, 272, 250] • Hybrid: [273]
AE • EC: [264, 274]
DBN • SI: [275]
GAN • EC: [276, 277, 278]
RL • EC: [61, 46, 279, 117] [119, 118, 280] [281, 282, 283] [121]

Figure 2.3: Taxonomy of the reviewed literature on Evolutionary Computation and Swarm Intelligence algorithms applied to the optimization of Deep Learning models. The taxonomy is structured by the domain of the Deep Learning model under focus (topology, hyper-parameters and trainable parameters), further discriminated by the specific Deep Learning model under consideration and the type of bio-inspired algorithm in use (*EC*: Evolutionary Computation; *SI*: Swarm Intelligence; *Hybrid*: a mixture of both).

better generalization properties while keeping its complexity to its minimum required. Originally applied to ANN models, different NE variants have been applied in the last few years to optimize DL models, not only in terms of their topology, but also jointly with their structural and training hyper-parameters (e.g. kernel size, activation function, dropout and learning rate). In some few cases, trainable parameters have also been considered in the set of variables to be optimized via NE [62]. Furthermore, since they resort to evolutionary algorithms at its core, NE approaches have stimulated over the years a manifold of other bio-inspired approaches, in a way to assess whether the same optimization problem can be tackled more effectively with alternative search strategies and operators.

All in all, in terms of topological optimization CNNs are arguably the most targeted DL models to date. CNNs' topology optimization is faced by scientific community in two ways; layer by layer or by blocks. In layer-wise optimization, hyper-parameters are fixed and networks are fully evolved using bio-inspired solvers, such as GA [79] and customized versions of other Evolutionary Algorithms [77]. In this last work, the so-called AmoebaNet-A model settled a state-of-the-art landmark score on the ImageNet dataset (83.9% accuracy), including comparisons to other search strategies (random search and RL). Another approach proposed in [94] resorts to PSO to optimize a block formed by dense layers. Once optimized, this block is stacked along with convolutional and pooling layers configured with fixed hyper-parameters, and ultimately used to address an image classification task. This work exemplifies a research trend focused on optimizing the topology of certain parts of the entire DL architecture, in an attempt at reducing the cardinality of the search space and speeding up the search process, at the cost of being much less exploratory in terms of network configurations than other counterparts. There is another important matter to be taken in account when performing topological optimization: the encoding of solutions, which impacts directly on the dimensionality of the search space. Actually, initial improvements of NE approaches were achieved thanks to novel network encoding strategies, which allowed for an easier exploration and less computational cost than preceding alternatives.

Another strand of literature has elaborated on more complex problem formulations by jointly addressing the optimization of the topology of the network along with its hyper-parameters. Again, CNNs have become central in related studies. An illustrative work is the one in [40], where an Evolutionary Algorithm (EA) is used with different mutation operators operating on topological variables, structural and training hyper-parameters such as the filter size, the convolution stride, learning rate or the insertion/removal of convolutional layers, among others. Studies in this field tend to be similar to each other in terms of the complexity of the optimization problem under consideration. Thus, new proposals are usually made by customizing the operators (mutator, selector) of optimization algorithms or by developing custom encoding strategies, as in [176] where a PSO variant is introduced based on an IPv4 based codification scheme with varying length.

Despite the predominance of CNNs in topological optimization, RNNs (and in particular, LSTM networks) have also been a subject of study in this research area, In [96] a Differential Evolution (DE) solver is proposed for achieving this purpose and efficiently undertaking a wind forecasting regression task. It is relevant to observe that when both architecture and hyper-parameters are evolved for RNNs, certain hyper-parameters are recurrently considered in related studies, such as learning rate, dropping frequency factor [190] or batch size [185, 184]. In general terms, the aforementioned DE appears to be the most applied meta-heuristics in LSTM. A few exceptions can be found, such as [190] (Bat Algorithm (BA)), [103] (ACO) and [185], where a comparison is made between DE, PSO and SA, concluding that DE reaches better performance levels. Hybrid approaches have been also explored for the optimization of RNNs, as in [106] where the architecture (i.e. connection pattern) is optimized by means of a hybrid PSO-GA solver. An interesting point arises when inspecting in detail this set of studies: the creation of custom objective functions to allocate different (usually conflicting) optimization goals. The work in [96] is an example of how a customized objective function can yield topologically optimized network designs that achieve a balance between performance and model complexity, being the latter of particular interest for the deployment of the model in resource-constrained embedded devices.

Other DL models have grasped a remarkable attention in topological optimization. AEs have been optimized topologically, often along with structural hyper-parameters in those cases where convolutional layers are involved. In their original formulation, AEs are composed by stacked dense layers (*encoder*) producing a low-dimensional representation of the input, which is reconstructed by another set of stacked dense layers (*decoder*). A contribution from 2015 [108] presented a way to generate promising AE architectures by mutating candidates by using a customized EA, whose mutation operator is based on the reconstruction error achieved by the decoder. Also in this vein, a mini-batch variant training method was proposed (*evo-batches*) aimed at reducing the computational cost when a large number of candidate networks have to be evaluated in large datasets. Decoder and encoder topologies of studies related to AEs are often assumed to be symmetrical [108]. However, in [107] a more flexible architecture was proposed, where the decoder is evolved along with the encoder and does not have to mimic its architecture. In this reference several operations were applied to topological variables, such as layer addition (random number of neurons), layer removal, application of Gaussian perturbation to the number of neurons, or layer swapping. To wrap up the activity noted in AEs, some notorious works are [196] and [191], where PSO and GA are respectively used to evolve topology and structural hyper-parameters of AEs comprising convolutional layers.

Proceeding forward with the analysis, the attention is focused on DBMs, whose architecture can be very similar to Deep Belief Network (DBN) in terms of the structural hyper-parameters involved in the optimization process. Given this similarity and the relative scarcity of studies related to these models, in what follows they are jointly analyzed. The majority of works related to the topological optimization of DBMs and DBNs pay special attention to the process of optimizing both architecture and some hyper-parameters. Moreover, most of them rely on Swarm intelligence algorithms, such as PSO or ACO. An exception can be found in [207] where ABC is used to optimize DBNs' structure, learning rate, momentum and weight decay. The results are compared to those yielded by other bio-inspired solvers: Firefly Algorithm (FA), Cuckoo search (CS) and Harmony Search (HS). FA for 2- and 3-layered DBN, and ACO for single-layer DBN, resulted to yield the best performing network architectures for the image reconstruction task under consideration. The rest of contributions consider a combination of all or some of learning rate, momentum or weight decay hyper-parameters, focusing the application of the optimized model to different practical problems such as traffic flow prediction [208] or the detection of turbine failures [210]. The work in [200] introduces a novel way to optimize structure and hyper-parameters of DBMs, and compares the performance of DBMs for image classification when optimized with different flavors of the PSO solver, random search and several HS variants. They concluded that bio-inspired techniques are suitable to optimize DBMs, beating Random Search in all considered datasets. Nevertheless, network topologies optimized via PSO and HS solvers scored similar performance levels. This last observation connects directly with one of the points remarked in the critical analysis of Section 2.4.

In terms of algorithmic variants, some hyper-heuristic techniques like [202] proposed an approach to optimise DBNs' structural hyper-parameters, i.e. number of hidden units, along with non-structural hyper-parameters like learning rate, and some hyper-parameters related to the heuristic algorithm (number of epochs or iterations). Hyper-heuristics have also been used to optimize CNNs, [232] presents a method to select the best heuristics, where batch size, number of epochs, neurons on the fully connected layer, dropout and learning rates, rho and epsilon factors are evolved.

There are also a few works dealing with the topological optimization of GANs. In [213], a meta-heuristic approach to evolve GANs' discriminator and generator is introduced. Specifically, a GA is used to evolve the architecture, activation functions of each layer and initialization mode in both generator and discriminator. Furthermore, an optimization of the loss function is done, taking them from a bunch of well-known formulations. Besides, training hyper-parameters are also mentioned in this work as potentially evolvable variables (yet not optimized in practice), such as the gradient-based solver used to learn the trainable parameters, the batch size and the number of epochs. Shortly thereafter, Costa et al. [114] proposed an approach to optimize the architecture and parameters of both the generator and discriminator modules of a GAN. The approach was based on DeepNEAT and adapted to the context of GAN optimization. Linear, convolutional and transpose convolutional layers were directly mapped to a phenotype representing the final network. In all layers the activation function was evolved, and in the case of convolutional and transpose convolutional layers, the output channels were also considered.

Finally, in the field of RL, the tendency observed in the literature analysis is to use NE approaches to optimize both topology and trainable parameter (weights) of the neural network mapping the output of the environments to the actions to be taken by the agent. Commonly, NEAT is used for this purpose [119, 117, 116, 118], which becomes in charge of optimizing the neural network involved in Deep RL approaches. A real-time adaptation of NEAT was used in [61] to evolve agents for the NERO videogame, placing an emphasis on the need for efficient workarounds to alleviate the complexity of neuro-evolution methods.

On a summarizing note, the literature on bio-inspired algorithms applied to architecture optimization has a long history departing from NE, which was originally applied to evolve ANN architectures. Since then, many other meta-heuristics have been applied to optimize architecture and hyper-parameters of DL models. Given that networks can have variable-length topologies, a good solution encoding strategy is essential to lessen computational costs and the time of execution without hindering the representability of all network configurations. Remarkably, modern bio-inspired solvers such as FA, BA and CS have been lately used with competitive results with respect to classical solvers (EA, PSO and ACO).

2.3.2. Hyper-parameter Optimization

Arguably, one of the optimization tasks where bio-inspired methods have been traditionally applied within the Machine Learning field is hyper-parameter tuning. It is well-known that hyper-parameter tuning usually yields better performance levels than off-the-shelf model configuration. When shifting the focus towards the hyper-parametric optimization of DL models, two major fields are spotted. On the one hand, literature focused on optimizing parameters related to the training algorithms, such as *learning rate*, *batch size* or *momentum*. On the other hand, architectural hyper-parameters, which are layer-type-dependent, e.g. filter size, number of kernels, activation functions or stride size in CNNs. Actually, given the high number of structural hyper-parameters of convolutional layers, CNNs have protruded as one of the most explored DL models for hyper-parameter optimization, in many cases jointly with training hyper-parameters such as the learning rate, momentum or weight decay of gradient solvers. Although there are some contributions focused exclusively on the optimization of structural hyper-parameters, the mainstream is to jointly address the optimization of topology and hyper-parameters, as the literature on topological optimization examined in the previous subsection has clearly revealed.

Let us start from 2016, when [233] proposed a set of bio-inspired meta-heuristics (BA, FA and PSO) to optimize the aforementioned hyper-parameters in a CNN used for Parkinson disease identification from image data. Likewise, [215] proposed a DE-based approach to optimize filter sizes, number of neurons in the fully-connected network end, weight initialization policy and dropout rate for sentiment analysis. Comparisons were made to GA and PSO, achieving better results in terms of accuracy and computational efficiency. The optimization of dropout probability is other common approach that some authors have tackled using different solvers, including CS, BA, FA and PSO [234] or hybrid GA and TS algorithms [236]. In this latter work random search and Bayesian optimization were proven to perform worse than the proposed hybrid meta-heuristic algorithm over the considered image classification datasets. Batch size and learning rate were also regarded as optimization variables.

Moving on to RNNs, very few papers are focused only on hyper-parameter optimization, conforming to the general trend observed in the analyzed literature. Dropout optimization is tackled for LSTM networks in [225] and [226], where ACO is used for engines vibration prediction. The connections between neurons are activated/deactivated to accomplish the task, which is very similar to the approach carried out in [227]. However, the aspect to be highlighted in this last reference is that a PSO is utilized to optimize the output connections of an Echo State Network (ESN), a randomization-based RNN model belonging to the family of Reservoir Computing models.

2.3.3. Trainable Parameter Optimization

In recent years the advent and progressive maturity of new parallel and distributed computing paradigms have reignited the interest of the scientific community in applying bio-inspired optimization algorithms to train DL models. Although each model type is different in terms of topology, they share some disadvantages resulting from the adoption of gradient back-propagation training methods, such as gradient vanishing/exploding and proneness to get stuck in local optima. Evidently, the complexity of the problem grows up as the number of parameters involved in the optimization process increase, yielding largely non-convex search landscapes. These acknowledged issues have been extensively studied by the community by proposing different workarounds. Nonetheless, an increasing trend towards the use of bio-inspired solvers for this purpose can be noticed in recent literature, as their search operators do not rely on gradient back-propagation anyhow, and therefore avoid its drawbacks effectively. Based on this rationale, the debate will delve into how the community has adapted different bio-inspired solvers for training DL models.

A first examination of the literature exposes two main tendencies followed by the community, which are reflected in the second level of the corresponding taxonomy branch of Figure 2.3:

- Approaches that combine bio-inspired solvers with traditional training algorithms, which aim to overcome the disadvantages introduced before. Almost the entirety of studies adopting this hybrid design strategy are focused on CNNs, implementing the aforementioned combination in many different ways. A straightforward way to overcome falling in local optima is to evolve an initial set of values for the trainable parameters (weights, bias) that sets the gradient back-propagation solver on a promising path towards the global minimum of the loss function. In [247] this approach is adopted for training a CNN using the ABC algorithm. Other works [239] combine GA and Stochastic Gradient Descent (SGD): GA evolves new candidates through its search operators, but the fitness function is evaluated after some training epochs of SGD. Similarly, in [248] PSO is used to evolve the trainable parameters of the last layer of a CNN, while the parameters of the rest of the layers are learned via SGD. Comparisons with the CNN trained exclusively with SGD rendered an enhanced convergence speed and final accuracy on image classification tasks. Last but not least, in [250] the CS algorithm is used to train RNNs following two strategies: one trained using only this solver, and the other combining CS and gradient back-propagation. A benchmark comparison to networks trained with conventional gradient back-propagation and different variants of the ABC algorithm discovered that all models trained using bio-inspired optimization techniques performed better than the RNN trained with gradient back-propagation.

- Approaches in which training is performed completely using bio-inspired optimization methods. Most references embracing this second strategy deal with RNNs and CNNs, and differ from each other mostly in the search algorithm being considered. All in all, a common approach is to evolve the trainable parameters of the model (via the search operators of the bio-inspired solver at hand), and evaluate it in terms of loss value or any other performance estimator linked to the task at hand. In this line, in [260] and [262] two SA-based solvers are proposed and assessed for optimizing the parameters of a CNN, achieving better performance scores and better convergence speed than the same model trained via gradient back-propagation. LSTM network training has also been tackled by using different bio-inspired optimization techniques. A good exponent is the work in [265], where HS, Gray Wolf Optimizer (GWO), Sine Cosine Algorithm (SCA) and Ant Lion Optimization (ALO) were compared to each other when used to learn the trainable parameters of different LSTM model configurations. It should be emphasized that despite the diversity of methods considered in this work, no comparisons to traditional training solvers were reported, uncovering one of the critical points discussed in Section Section 2.4.

Before proceeding with this critical analysis, some comments on Deep RL models are provided. Bio-inspired algorithms have been lately postulated as efficient alternatives to solve several optimization problems underlying these models. A first approach is to train parts of the architecture via evolutionary algorithms, and the rest using SGD. This is indeed what the study in [251] proposes: a Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is used to evolve weights for the behavior-generating network, while the rest of the Deep RL architecture (a convolutional AE) is trained by using a gradient based method. This work continued the research line started years before in [280], where CMA-ES was used to train simpler RL networks. All in all, Evolutionary Algorithms have largely demonstrated to be efficient solvers to train Deep RL models, as shown in renowned studies such as [46] (GA with Novelty Search) and [282] (DE and Novelty Search). Recent works [281] have also explored the capabilities of multitask optimization evolving multiple related RL tasks at the same time, taking advantage of the transfer of genetic information between tasks. In other works, NEAT is used to optimize the trainable parameters and the topology of a Deep RL model [119, 117]. On the other hand, in [256, 257] an hybrid EA algorithm is proposed, where a population of networks is trained and periodically evolved, exploiting Lamarckian transfer capabilities. This same procedure is used in [258] as a way to inject information about the gradient in the population of individuals maintained by the evolutionary algorithm during the search. Other hybridization techniques consist of the use of different networks in the same architecture, where some of them are trained via SGD, and others evolved with evolutionary operators. This is the case of [259], where the parameters of an RNN used for determining the actions of an agent are evolved using Cooperative Synapse Neuroevolution (CoSyNE), an evolutionary algorithm that enforces subpopulations at the level of a single trainable parameter. This work built upon the findings in [279] and extrapolated them to complex Deep RL models, showcasing how evolutionary algorithms can evolve small networks capable of reaching competitive performance levels.

2.4. Critical Methodological Analysis

The above corpus of reviewed literature sheds evidence on the vibrant activity of the intersection between bio-inspired optimization and DL. So far the community has reported interesting findings in what refers to hyper-parameter optimization, topological search and small/medium-sized network training. Notwithstanding this noted activity, the critical analysis of these contributions has disclosed a number of poor practices and methodological shortages that should be underlined to set them down in black and white. A brief discussion about these issues is carried out next, settling the necessary rationale for the experimental part of this chapter.

2.4.1. Lack of Benchmark Datasets and Tasks

A major problem observed in the literature is the heterogeneity of datasets used to validate new algorithmic approximations for the optimization problems under analysis. Even when the task is clearly defined (e.g. image classification or time series forecasting), the possibility to compare the results obtained by different studies becomes unfeasible since the considered datasets are not the same.

For the community to gain verifiable evidence about the claimed gains of upcoming proposals, consensus should be reached about the datasets/tasks that should be utilized for comparison purposes in the future. Unfortunately, the diversity of datasets/tasks over which some of the new contributions are assessed seem to go in the opposite direction, calling into question whether the reported performance improvements can be extrapolated to other learning problems.

2.4.2. Unrealistic Complexity of Deep Learning Models

Besides the heterogeneity of datasets/tasks discussed above, it is often the case that the DL models under consideration do not meet the complexity levels of the state of the art for the task under consideration. This is particularly concerning in works related to model training (Problem 4), where the cardinality of the search space faced by the meta-heuristic solver is in the order of several thousands to millions of optimization variables (trainable parameters). For instance, a recent work on image classification using the well-known MNIST dataset has recently established a new record in accuracy (0.1% test error rate) with a model comprising 57.02 millions of trainable parameters [284]. However, most of the reviewed literature on training via bio-inspired solvers rarely considers DL models that surpass a few thousands of trainable parameters.

This issue again calls for a major reflection on whether research advances are missing the real challenge underneath the use of bio-inspired algorithms in such large search spaces (scalability, exploitation of the correlation among decision variables).

2.4.3. Comparison Methodology

Even if addressing and effectively solving the preceding two issues, several methodological aspects still remain often overseen when comparing among different solvers for a given task/dataset/optimization problem scenario:

- Baseline schemes: the literature analysis revealed a fraction of contributions discussed on extensive experiments with several new meta-heuristic algorithms for a given optimization problem, without including in the benchmark standard solvers utilized in the past for the same problem. This, again, is particularly worrying in regards to Problem 4 (model training): comparisons should compulsorily include gradient back-propagation based solvers widely used for the same purpose (e.g. SGD, Adam). Overlooking the analysis of whether bio-inspired algorithms perform competitively with respect to established solvers for the same purpose is counterproductive for the potentiality of this research area.

- Solver's complexity: in regards to the previous point, a major consensus should be reached on the dimensions over which comparisons among solvers should be made for conclusions to be fair and of practical value. It is particularly concerning that no computational complexity assessment has been made in almost all studies reported to date. Instead, the focus is placed on the predictive performance gains yielded by the newly developed solver with respect to its counterparts in the benchmark. Unless properly quantified, considered and eventually alleviated by virtue of new research directions, the huge computation cost of EDL can be an obstacle for benchmarks to lead to conclusions of practical relevance. Evolutionary Computation and Swarm Intelligence algorithms rely on search space sampling and model evaluation, both of which are not efficient strategies for large-scale applications.

- Objective function(s): it can be noticed that relevant divergences emerge in how the optimization algorithms proposed over the years are guided when attempting to solve a given optimization problem. For instance, a common practice is to reserve a validation data subset over which a measure of performance related to the task at hand is computed (e.g. accuracy in image classification). This measure is used as the objective function guiding the search of the proposed optimization algorithm. However, depending on whether this validation subset is kept fixed or shuffled, a partition bias might affect the generalization capabilities of the evolved network, specially when dealing with small datasets.

  On a similar reasoning, when dealing with imbalanced datasets the standard definition of accuracy is known to be not adequate to quantify the performance of the model in the minority class, and could exacerbate further the aforementioned problems. In what refers to Problem 4 (trainable parameter optimization), this issue becomes even more serious because derivatives of the loss function are not needed any longer, hence widening the portfolio of possible objective functions. To date, there is no clear answer whether differentiable loss functions should be selected at the objective function of bio-inspired optimization algorithms used for model training, or, instead, alternative task-dependent objectives should be formulated.

- Parameter tuning of solvers: additional issues arise in how different solvers are compared to each other for a given optimization problem. To begin with, it is often the case that no evidence is provided about the optimality of the parameters controlling the behavior of the search algorithm itself. In the research community working in bio-inspired optimization, it is largely accepted that a good parameter tuning is crucial for ensuring fair comparisons among algorithms [285]. Since the objective function evaluation of candidates in problems related to DL is usually costly in terms of computational effort, the parameters of the bio-inspired algorithm are often set equal to values retrieved from past works, or conforming to common practice. This unfairly biases the discussion, usually leaving unclear whether the reported performance gaps are incidental. The research record noted around the usage of hyper-heuristics [202, 232] avoids to an extent this comparison bias, but the problem still prevails in most works proposing new bio-inspired methods.

- Assessing the impact of randomness on the results: another methodological aspect that has not been properly considered in most related literature is the fact that several sources of randomness can collide into an optimization problem. For instance, in Problems 1, 2 and 3 not only the search algorithm comprises a number of stochastic operators, but the training algorithm in use can also induce randomness in the obtained results. For instance, the values of trainable parameters optimized by the SGD solver depends on the composition of the mini-batches over which gradient estimates are computed. Such mini-batches comprise a number of examples from the training set, which are usually shuffled between successive epochs. This source of randomness could justify training the optimized network several times (*runs*) and aggregating the results for a more reliable fitness computation. Otherwise, this should be conceived as an additional factor motivating a proper statistical assessment of the significance of performance gaps between different optimization algorithms, adding to the randomness induced by their search operators.

Surprisingly, only a few exceptions have embraced the usage of statistical tests for this purpose, leaving most of the experiments reported in this field dubious and inconclusive. Furthermore, experiments with several datasets, tasks and optimization algorithms should embrace the methodological practices for multiple comparisons deeply rooted on the scientific community, such as critical distance plots [286] or Bayesian tests [287].

- Reproducibility of results: although this is a claim that emerges in almost any field of research, the need for reproducibility becomes particularly pressing in this area. Reasons go far beyond the verification of the contribution reported in emerging studies: the community can expedite the achievement of novel advances in the field if the software and experimental results of preceding works are shared within the community. The current status in this matter is not that concerning, as many open-source software frameworks exist with the functionalities required to develop new approaches and experiments (see Table 2.1). Unfortunately, many contributions published lately still do not provide any access to the software implementing their proposals. When given, it is also frequent to see that the source code is made ad-hoc for the problem at hand, thereby dilating the time needed for building new proposals on top of existing ones.

2.4.4. Software Implementations of Limited Practical Utility

Nowadays, DL architectures scoring competitively in tasks defined over real-world datasets are usually composed by millions of trainable parameters. When addressing Problem 4 (trainable parameter optimization), the huge search space faced by the optimization algorithm under consideration requires intelligent means to exploit the relationships existing among the optimization variables. Actually, gradient based solvers adopt this strategy by the back-propagation of the gradients throughout all layers compounding the DL model, which gives rise to an implicit mechanism to exploit the correlations between the optimization variables. In this context, there is an entire research area dedicated to the large-scale global optimization, plenty of algorithmic proposals where synergies among optimization variables are exploited by assorted means. Nonetheless, many works still revolve on the naive application of standard bio-inspired solvers, neglecting such interactions between variables.

Further along this line, it should be noted that the vibrant activity of the area is not in accordance with the ultimate goal for which bio-inspired algorithms are being proposed for training DL models. As told by the currently available implementations in the literature, the computational cost of population-based meta-heuristics is enormous, and yields far longer training times than off-the-shelf gradient back-propagation approaches. Even if the software implementation of algorithmic proposals reported to date may have been restricted to experimental settings, a complexity analysis should have been performed to accounting for both their benefits and drawbacks, so that the community can delimit realistic boundaries for the practical utility of these advances.

2.4.5. Metaphor-based Publication Series

Last but not least, we emphasize on the claims of recent studies about the justification of new meta-heuristic algorithms just by the biological metaphor in which it is allegedly inspired [19]. In the conducted literature review there are several publication series in which the same optimization problem were tackled by different bio-inspired solvers within a short time period. By no means these contributions provide any scientific value for the general knowledge of the field, even if publication workflows run at different speeds.

Disregarding the reasons for these practices, results with optimization algorithms inspired by different biological behaviors and phenomena should not be shattered over different short-elapsed publications. They can provide much more valuable insights when presented and discussed together.

2.5. Empirical Analysis of Evolutionary Algorithms for Training and Designing Neural Networks

Complementing the critical literature analysis offered in the previous section, we now pause at an experimental setup designed to unveil strengths and weaknesses of EDL when faced with complex models and realistic network sizes. Two cases of study are herein considered: one for achitecture and hyperparameter optimization (Problem 1 and Problem 2), and another one for training parameter optimization (Problem 4), covered in Sections 2.5.1 and 2.5.2, respectively.

2.5.1. Case of study I: Architecture and Hyper-parameter Optimization of Deep Learning Models with Bio-inspired Algorithms

In this first case of study the aim is to optimize topology and structural hyper-parameters of DL models. This, has been an open challenge in the last few years. In particular, this experimental study focuses on finding the best CNN architecture along with their structural hyper-parameters for solving image classification problems. Two recent frameworks are considered:

- EvoDeep [50], an evolutionary algorithm specially designed to optimize both hyper-parameters and architecture of a DL model, selecting the type and size of layers compounding the evolved architecture.
- AutoKeras [55], is another modern AutoML system built on top of the renowned Keras Python library, which is able to automatically generate highly-optimized neural network.

Taking into account the recent activity noted in this area, herein a comparison and discussion on the performance of these two consolidated frameworks for topology and hyper-parameter optimization of DL models is carried out. The comparison is made using three important measures: accuracy, time and model complexity. In what follows, EvoDeep (Section 2.5.1.1) and AutoKeras are introduced (Section 2.5.1.2) and the designed experimental setup (Section 2.5.1.3) is presented. Finally, the outcomes are discussed in Section 2.5.1.4.

2.5.1.1. EvoDeep: Evolutionary Computation for Deep Neural Network Topology and Hyper-parameter Tuning

EvoDeep is an evolutionary framework based on an EA that follows a $(\lambda+\mu)$ strategy, where λ indicates the number of new individuals produced at each generation, and μ represents the number of individuals selected for the next generation. Each individual of the population is a network architecture with its respective hyper-parameters. The fitness for each individual is the accuracy of the neural network when solving a classification problem. At every generation, the recombination and mutation operators are applied to generate a new individual for the next generation.

EvoDeep finds increasingly better neural network architectures for CNN using elementary convolutional and fully connected layers. Specifically, EvoDeep evolves a network by using only the original data and an set of permitted layers (i.e. `Convolution2D`, `Flatten`, `MaxPooling2D`, `Reshape`, `Dense` and `Dropout` as per Keras notation). The number of layers represents the depth of the evolved network, and its search range is set to $[3, 20]$ with a step size of 1 layer. Moreover, the training hyper-parameters can be also specified in its configuration. These parameters are as follows:

- Optimizer, chosen among `Adam`, `SGD`, `Rmsprop`, `Adagrad`, `Adamax` or `Nadam`.

- Number of epochs: an integer value in the range $[2, 20]$, with a step size of 2 epochs.
- Batch size: this value has a range of $[100, 5000]$ with a step size of 100 examples.

In terms of data input, EvoDeep requires a two-dimensional matrix where the number of rows matches the number of examples of the database and the number of columns the size of the images (product of width, height and channels).However, EvoDeep is mostly used to optimize grayscale images (one channel) rather than RGB (three channels). Table 2.3 shows the configuration of EvoDeep's internal EA used for this experiment.

Table 2.3: Parameter configuration set for the EvoDeep framework.

Parameter	λ	μ	`cxpb`, p_c	`mutpb`, p_m	`newpb`	`ngen`, n_{gen}
Description	Num. of newly produced individuals per generation	Num. of selected individuals for the next population	Crossover probability	Mutation probability	Probability of adding a new layer to the network	Num. of generations
Value	10	5	0.5	0.5	0.5	20

2.5.1.2. AutoKeras: an AutoML Reference

AutoKeras is a software that also allows to find the best neural network architecture for a given data set and task [55], offering several search engines for this purpose. In version 1.0.2 the following search methods are considered:

- Random: it performs a random search of the models in relation to their depth and layer type.
- Greedy: it groups the parameters into several categories. For each category, the tuner uses a greedy strategy to generate new values for the hyper-parameters and to generate new values for one of the categories of hyper-parameters. It uses the best trial so far for the rest of hyper-parameter values.
- Hyperband: departing from a given model, this bandit-based algorithm searches the best hyper-parameter values for this model by running several random configurations for a scheduled number of iterations, using earlier results to retain good candidate configurations that are evaluated for longer runs.
- Bayesian optimization: the search space is explored using morphing. This optimization method is based on the three basis of Bayesian optimization: update, generation and observation.
- Task-specific: AutoKeras tries with a task-specific tuner of the model, and evaluates the most commonly used models for the task at hand, which in

the case at hands is image recognition. This is the default configuration. AutoKeras has two trial blocks: Vanilla and ResNet. The best model is taken from one of these two.

2.5.1.3. *Experimental Setup*

For this experiment, 4 diverse and representative data sets are considered according to their complexity: Horses or Humans (HORSEHUMAN [288]); the `vangogh2photo` dataset from the so-called `cycle_gan` repository (VANGOGH [289]), MNIST [290] and CIFAR-10 [291]. On the one hand, Horses or Humans and Van Gogh or Photo have two classes, but Van Gogh or Photo is unbalanced (only 400 images belonging to the minority class). On the other hand CIFAR-10 and MNIST are more complex databases in terms of number of examples and classes: both databases have 10 different classes. Moreover, for this experiment both color (RGB) and grayscale versions of these datasets are considered so as to assess the influence of the color space in the complexity and performance of the models evolved by the compared frameworks. Other contributions in the literature have considered datasets of larger scales, such as CIFAR-100 [80, 136, 138, 292, 151, 146] or ImageNet [77, 172] (the latter transferring knowledge from models learned from CIFAR-10 and CIFAR-100 beforehand). Nevertheless, the datasets and experiments designed in this first case of study are illustrative enough of the potential and caveats that EDL still undergo when applied to the optimization of topology and hyper-parameters.

Table 2.4: Datasets used in Case of Study I.

Dataset	**Shape**	**# classes**	**# Instances (train/test)**	**Characteristics**
HORSEHUMAN	$300 \times 300 \times 3$	2	1,027 / 256	Balanced dataset, binary classification, RGB
HORSEHUMAN-G	$300 \times 300 \times 1$	2	1,027 / 256	Balanced dataset, binary classification, grayscale
VANGOGH	$256 \times 256 \times 3$	2	6,687 / 1,151	Imbalanced dataset, binary classification, RGB
VANGOGH-G	$256 \times 256 \times 1$	2	6,687 / 1,151	Imbalanced dataset, binary classification, grayscale
CIFAR-10	$32 \times 32 \times 3$	10	50,000 / 10,000	Balanced dataset, multi-class classification, RGB
CIFAR-10-G	$32 \times 32 \times 1$	10	50,000 / 10,000	Balanced dataset, multi-class classification, grayscale
MNIST	$28 \times 28 \times 1$	10	60,000 / 10,000	Balanced dataset, multi-class classification, grayscale

In order to account for the statistical behavior of the frameworks under comparison, 5 independent runs of AutoKeras are carried out, EvoDeep and random models, from which the best model in terms of its accuracy in validation is selected. Random models are obtained using EvoDeep without the evolutionary process. The number of tested individuals is the number of evaluations that EvoDeep would make via its evolutionary strategy. The training phase has been done by splitting the training set as follows: 80% for training and 20% for validation. After that, the model is trained again with the whole training set, and evaluated on the test set.

2.5.1.4. *Results and Discussion*

Table 2.5 shows the accuracy scores obtained for the models evolved with EvoDeep, AutoKeras and random search over the color and grayscale datasets under consideration. The best results for each dataset are highlighted in bold. On the one hand, it is straightforward to observe that EvoDeep outperforms random models over both train and test datasets. However, AutoKeras still offers a better performance than EvoDeep. AutoKeras features the best test results in each dataset. To sum up, the results of EvoDeep are close to AutoKeras, which it is one of the best AutoML tools in this area. In Vangogh or Photo and MNIST the difference between them is less than 1% in test and 2% in Horses or Humans. Therefore, EvoDeep shows a great performance on these databases.

When shifting the scope towards the results obtained for the grayscale databases, similar conclusions can be drawn. In relation to the HORSEHUMAN dataset, the results issued by EvoDeep are similar to the previous ones in terms of accuracy over test, whereas AutoKeras improves its performance by 3%. The results for CIFAR-10-G are worse when compared to the color ones. The exception in this trend is VANGOGH-G, due to the bad results obtained by the random models and EvoDeep. The accuracy in test decreases approximately 17% in EvoDeep and 27% in random models. By contrast, the results obtained for this dataset by AutoKeras are similar to the ones obtained for its colored counterpart.

Table 2.5: Results in terms of accuracy for color datasets and algorithms (in %).

Color datasets	**HORSEHUMAN**		**VANGOGH**		**CIFAR-10**			
	Train	Test	Train	Test	Train	Test		
Random	89.19	89.06	98.10	92.96	73.06	53.63		
EvoDeep	97.07	89.40	**100.00**	98.87	**99.99**	60.55		
AutoKeras	**100.00**	**91.40**	**100.00**	**99.48**	95.28	**73.26**		
Grayscale datasets	**HORSEHUMAN-G**		**VANGOGH-G**		**CIFAR-10-G**		**MNIST**	
	Train	Test	Train	Test	Train	Test	Train	Test
Random	94.06	89.84	94.02	65.24	73.44	49.35	99.87	98.40
EvoDeep	**100.00**	89.84	96.66	81.58	85.52	56.79	**100.00**	98.69
AutoKeras	**100.00**	**94.53**	**100.00**	**99.30**	**94.42**	**71.67**	99.99	**99.40**

After the comparison of EvoDeep, AutoKeras and random models in terms of accuracy, the execution time of the previous results, in minutes, are examined (Table 2.6). A first inspection of the results in this table reveals that AutoKeras has the best performance with lower times than EvoDeep. Note that EvoDeep stops if no improvement is made over 5 consecutive generations. This is what occurs in MNIST: EvoDeep needs more computation time than AutoKeras and random models, but its accuracy results lie in between. This fact makes EvoDeep a reliable software because, even though it needs more computation time, the quality of the models in terms of predictive accuracy are close to those of AutoKeras in most cases. When it comes to grayscale datasets, in general the computation time is lower than the time taken by the experiments with color datasets. Runtimes of AutoKeras are almost the same except for HORSEHUMAN-G, in which the time is approximately 2.6 times faster than that of HORSEHUMAN. EvoDeep and random models require less computation time in all the cases when compared to the colored ones. For example, VANGOGH-G is around 2.25 times faster than the previous experiments.

Table 2.6: Results in terms of time (minutes) for all datasets and frameworks.

Color	**HORSEHUMAN**	**VANGOGH**	**CIFAR-10-G**	
Random	**5.5**	**11**	**92**	
EvoDeep	22.0	50	322	
AutoKeras	13.5	14	110	
Grayscale	**HORSEHUMAN-G**	**VANGOGH-G**	**CIFAR-10**	**MNIST**
Random	**3.08**	**7.14**	**43.44**	**46**
EvoDeep	13.17	22.25	295.99	230
AutoKeras	5.19	13.98	109.98	262

Proceeding with the analysis of Table 2.7, where random, EvoDeep and AutoKeras are compared in terms of resulting model complexity (i.e. Number of layers and the number of trainable parameters of the best model). As proven by the results included in this table, random model produces very similar network architectures across the colored datasets: they all comprise 5 layers with varying size (between 10^3 and $2 \cdot 10^6$ trainable parameters). EvoDeep beats AutoKeras in terms of number of layers for 3 datasets, and in terms of the number of trainable parameters for 2 datasets (HORSEHUMAN and MNIST). The best model for HORSEHUMAN is achieved by EvoDeep, comprising a simple neural architecture with 3 layers and approximately $1.9 \cdot 10^6$ parameters. AutoKeras' model for the VANGOGH dataset has 9 layers and fewer trainable parameters in comparison to the models produced by the other frameworks. EvoDeep produces a model with only 5 layers for CIFAR-10, but AutoKeras' model has less parameters. Finally, for the MNIST dataset, EvoDeep discovers a better model than AutoKeras in terms of model complexity: fewer layers and parameters. To sump up, in terms of model complexity EvoDeep can be declared to perform better than AutoKeras in some of the considered colored datasets.

Table 2.7: Results in terms of model complexity for all datasets and algorithms.

Color datasets	HORSEHUMAN		VANGOGH		CIFAR-10	
	Layers	Parameters	Layers	Parameters	Layers	Parameters
Random	5	**179,922**	**5**	1,158,402	**5**	2,270,100
EvoDeep	**3**	1,895,012	9	18,756,012	**5**	10,821,230
AutoKeras	194	23,566,856	10	**31,944**	10	**144,849**

Grayscale datasets	HORSEHUMAN-G		VANGOGH-G		CIFAR-10-G		MNIST	
	Layers	Parameters	Layers	Parameters	Layers	Parameters	Layers	Parameters
Random	**4**	853,322	**3**	318,372	**5**	239,650	**5**	**211,245**
EvoDeep	7	**129,182**	6	783,352	12	1,883,220	8	5,409,980
AutoKeras	194	23,560,580	10	**31,364**	10	**144,269**	194	23,579,021

When analyzing the complexity of models discovered for the grayscale datasets, the results in Table 2.7 unveil a huge improvement in the number of parameters in random models and EvoDeep. Results by AutoKeras remain very similar to those for the colored datasets, with minimal differences in terms of the number of parameters. Regarding to the number of layers, random models have almost the same amount as colored ones. However, more remarkable changes are noticed for EvoDeep. The models discovered for the HORSEHUMAN-G dataset has more layers, but significantly less parameters. The same statement can be said for CIFAR-10-G. The results for the VANGOGH-G dataset are the exception: the best model has fewer layers and fewer parameters. These observations support the aforementioned claims on the complexity of models produced by AutoKeras with respect to EvoDeep and Random.

The previous experiments have shown some differences between the color and the grayscale databases. In fact, when focusing the analysis on the best model found by EvoDeep across all datasets, remarkable differences arise between the test accuracy and the complexity of such models corresponding to color and grayscale datasets. As summarized in Table 2.8, accuracy results are similar in both cases. The accuracy achieved by the best EvoDeep model over CIFAR-10 in higher than that of its grayscale version (CIFAR-10-G). The case with the largest difference in terms of accuracy is VANGOGH, which has a 17% gap. In terms of complexity, the best model of each grayscale database has fewer neurons than the corresponding colored dataset. In HORSEHUMAN-G, the model has approximately 14.69 times fewer neurons than that of HORSEHUMAN. For VANGOGH and CIFAR-10, this ratio gets close to 24 times and 5.75 times, respectively.

Table 2.8: Results in terms of accuracy and complexity for the best model encountered by EvoDeep for both color and grayscale datasets.

	HORSEHUMAN	HORSEHUMAN-G	VANGOGH	VANGOGH-G	CIFAR-10	CIFAR-10-G	MNIST
Test acc. (%)	89.45	89.94	98.87	81.58	60.55	56.79	98.69
# of neurons	1,895	129	18,756	783	10,821	1,883	5,409

Now, the main conclusions drawn from this discussion are summarized, leaving a further elaboration on the general lessons learned in regards to topology and hyper-parameter optimization for Section 2.6.2:

- EvoDeep has similar results to AutoKeras in the color databases HORSE-HUMAN, VANGOGH and CIFAR-10. Nevertheless, the computation time that EvoDeep requires is much higher than the one taken AutoKeras during its search. Considering the grayscale datasets, the difference between AutoKeras and EvoDeep increases. In particular, accuracy gaps over the VANGOGH dataset is particularly large: 81.58% in VANGOGH-G and 98.87% in VANGOGH. In terms of accuracy, EvoDeep performs better with the colored databases.
- Taking a closer look at the results in terms of accuracy and model complexity, EvoDeep needs less computation time in the grayscale datasets. Furthermore, this statement also holds in terms of model complexity. Although there are some models that comprise more layers when dealing with grayscale datasets, the number of total trainable parameters for all the models is much lower. All these facts contribute to a better overall performance of EvoDeep with grayscale databases.

To summarize, although AutoKeras has better performance in terms of accuracy in all datasets, EvoDeep gives competitive results and requires less computation time for simple datasets (i.e. MNIST). For several datasets, EvoDeep produces models with fewer parameters, while AutoKeras yield very complex models that may be unsuitable for application scenarios with stringent memory restrictions. In other datasets, the simplicity of the layers supported by EvoDeep enforces a higher number of neurons (and parameters) than AutoKeras for the same performance level.

The empirical results reported in this first case of study underscore the potential of Evolutionary Algorithms for the topological and hyper-parameter optimization of CNN networks, suggesting several possible improvements to frameworks appearing in the literature in forthcoming years. First, frameworks should incorporate sophisticated layers at their core, so that they become eligible for the evolutionary algorithm in use and ultimately lead to a reduced overall complexity of the optimized models. The overly complex models encountered by EvoDeep in some of the considered image classification datasets is a clear evidence that, for the sake of fair comparisons, all frameworks should ensure that the search spaces explored by their optimization engines are comparable to each other as well. Another possible improvement is to formulate topological and hyper-parameter optimization as a multi-objective problem, embracing as conflicting objectives the accuracy of the model, and a measure of its complexity (layers, parameters, computation time, etc). This output could allow the community to examine the behavior of new frameworks and solvers in regards to the performance and complexity of their produced solutions, making their output more flexible to implement the discovered models by taking into account both objectives. Later in the chapter we will revolve on these research directions on Section 2.6 and Section 2.7.

2.5.2. Case of Study II: Training Deep Learning Models with Bio-inspired Algorithms

This second case of study aims to shed light on the performance of bio-inspired optimization algorithms when applied to model training (Problem 4). The ultimate goal is to check whether bio-inspired algorithms can be a competitive alternative to gradient-based methods when undertaking image classification tasks over well-known datasets, ensuring that DL architectures of realistic complexity are in use. To this end, the experimental setup is designed to provide an informed response to the following research questions (RQ):

- RQ1: Should bio-inspired solvers exploit the layered structure of DL models during the search?
- RQ2: Do bio-inspired solvers perform competitively with respect to gradient-based solver for trainable parameter optimization, in terms of predictive accuracy and computational efficiency?

In the remainder of this second case of study, Section 2.5.2.1 introduces the evolutionary algorithm selected for the experiments, underscoring several changes made to its original definition to make it better suited to the optimization of trainable parameters. Subsection Section 2.5.2.2 provides further details on the experimental setup, including the DL models and datasets under consideration. Subsections Section 2.5.2.3 and Section 2.5.2.4 discuss in depth on the results obtained from the experiments, with a focus on RQ1 and RQ2, respectively.

2.5.2.1. SHADE-ILS: A Reference Evolutionary Algorithm for Large-Scale Global Optimization

Given the large number of variables to be optimized, SHADE with Iterative Local Search (SHADE-ILS) is selected as target evolutionary algorithm the experiments. SHADE-ILS is a renowned large-scale global optimization algorithm that has won recent international competitions in the field [33]. In particular, SHADE-ILS resorts to the global exploration capability of an adaptive variant of the DE algorithm (SHADE), which is helped by two local search methods, namely, a limited-memory version of the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, and multiple trajectory search that improve the candidate solutions encountered during the search. At every iteration, SHADE applies evolutionary operators on a population of individuals, followed by the application of one of the local search methods to the best individual of the population. The selection of which local search to apply is driven by the best expected relative improvement of each of the considered local search operator, which is given by the results produced by each local search alternative during recent generations. Moreover, SHADE-ILS incorporates a restart mechanism to avoid stagnation. These key algorithmic aspects and the excellent results scored in benchmarks and competitions make SHADE-ILS one of the baseline algorithms for large-scale global optimization problems, just like the one addressed in this second case of study.

Several changes have been done to the original SHADE-ILS algorithm to make it better suited to the optimization of the trainable parameters of a DL model. The objective function to be minimized over the search is the same than that used by gradient back-propagation approaches for the same model, namely, binary cross-entropy for binary classification and categorical cross-entropy for multi-class classification. Both are measured for all examples belonging to the training set. Secondly, the layer-wise structure of DL models is taken in account by devising several scheduling strategies. Each strategy establishes the criteria, order and number of generations for which the optimization variables (trainable parameters) belonging to each layer are evolved via the global search operators and local search techniques of SHADE-ILS. The design of the above strategies is motivated by RQ1, where the focus is placed on whether the layered structure of DL models should be exploited anyhow by the bio-inspired solver.

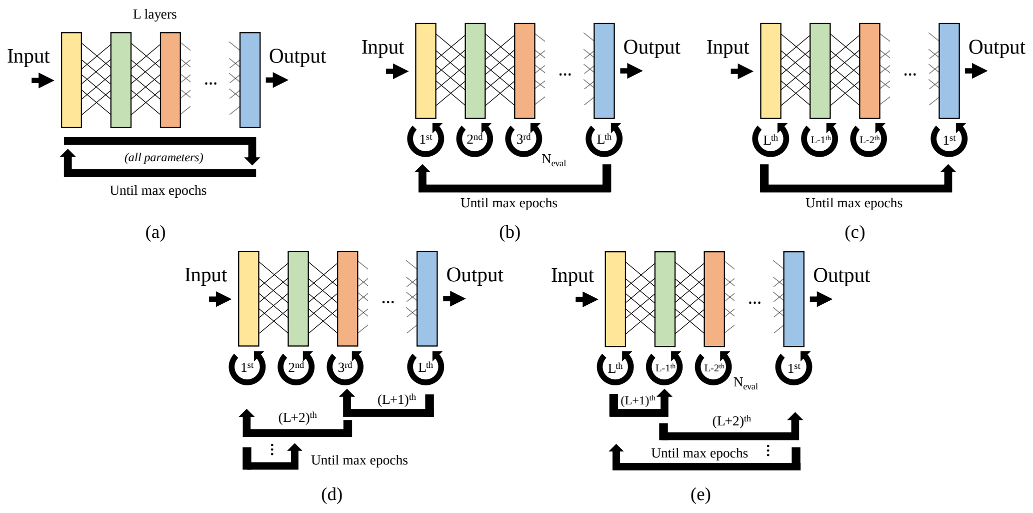

Figure 2.4: Diagram showing the different scheduling strategies proposed for optimizing Deep Learning models with the SHADE-ILS algorithm: (a) FULL-SHADE-ILS; (b) DOWN-SHADE-ILS; (c) UP-SHADE-ILS; (d) A-DOWN-SHADE-ILS; (e) A-UP-SHADE-ILS.

Specifically, the strategies considered in the simulations are as follows (see Figure 2.4 for a visual explanation):

- FULL-SHADE-ILS: all trainable parameters are optimized jointly, without considering the layered structure of the model to be trained. This is actually the strategy followed by most contributions to the literature dealing with the application of bio-inspired optimization techniques for DL models. As will be later shown, this strategy only works for network architectures of relatively limited size.

- DOWN-SHADE-ILS: trainable parameters are optimized starting from those belonging to the first layer of the network. Such parameters are evolved via SHADE-ILS for a certain number of generations, keeping the values of the remaining parameters fixed to their Glorot-based initialized values. The optimization schedule is repeated in order from the first to the last layer of the network for a maximum number of epochs.
- UP-SHADE-ILS: this schedule is similar to the previous DOWN-SHADE-ILS strategy, but departing from the last layer of the network, and proceeding upwards until the first layer of the network.
- A-DOWN-SHADE-ILS: an automated variant in which after a first iteration of the DOWN-SHADE-ILS optimization strategy, a relative improvement ratio of the network predictive accuracy is computed for every layer optimization step. This ratio is used to select which layer to optimize in subsequent epochs, so that layers whose optimization yielded larger improvements in the last epoch are more likely to be selected for optimization.
- A-UP-SHADE-ILS: this last strategy is similar to A-DOWN-SHADE-ILS, the difference being that the first layer-wise application of SHADE-ILS is done from the last to the first layer of the network (namely, under the DOWN-SHADE-ILS strategy). Once this initial stage is completed, SHADE-ILS proceeds analogously to the previous strategy, automatically selecting the layer with most potential margin of improvement.

When addressing RQ2, it is important to stress on the complexity of ensuring a fair comparison between gradient back-propagation solvers and bio-inspired optimization algorithms. The main reason is their essentially different search behavior. On one hand, gradient back-propagation approaches maintain a single solution to the problem, which is enhanced over epochs by exploit the mathematical relationships between the optimization variables (trainable parameters) through the tailored computation of their gradients. However, the search operator of gradient back-propagation techniques is simple, yet effective (gradients are personalized for every single variable) and efficient (gradient computations are highly parallelizable). By contrast, most bio-inspired algorithms rely on a population of individuals, which are evolved jointly by means of a series of search operators, so that the best individuals survive between generations. In summary, comparisons between both approaches should be fair not only in terms of accuracy performance, but also in terms of computational complexity.

For ensuring fairness in these terms, a straightforward decision is to define how *epoch* and *generation* relate to each other. First of all, it is clear that both concepts indicate when a full update of the network's parameters is complete: in gradient back-propagation, an epoch implies the application of a number of gradient updates to the whole network parameters. The number of updates depends on the size of the training set and the chosen batch size. Given that an epoch is defined as the optimization of all parameters composing the model, in SHADE-ILS an *epoch* corresponds to the optimization of all layers of the network under any of the strategies described previously. Assuming N_{eval} evaluations of the network per layer and L layers, an epoch for SHADE-ILS will comprise $L \cdot N_{eval}$ total evaluations of the network per epoch. Each evaluation of the network involves predicting the entire N_{train}-sized training set and computing the loss function. As a result, in general the number of evaluated training instances per epoch differs between SHADE-ILS ($N_{train} \cdot N_{eval} \cdot L$) and gradient back-propagation (N_{train}). Nevertheless, the experiments are conducted disregarding this issue, and is analyzed if SHADE-ILS, even if endowed with more computational budget per epoch, can beat the accuracy of networks optimized via Adam, one of the most renowned gradient back-propagation solvers.

2.5.2.2. *Experimental Setup*

The above two questions are tackled by considering 6 datasets for image classification: Hand gesture recognition (HANDS, [293]), Blood Cells classification dataset (BCCD, [294]), MNIST [290], Fashion MNIST (F-MNIST) [295], GTSRB [296] and CIFAR-10 [291]. Details of these datasets are given in Table 2.9. Given the amount of computational resources required to complete the experiments, a subset of the examples for each dataset is considered. For the same reason and except for the WBC dataset, images have been converted to grayscale to reduce the number of channels of the input image. Even with these simplified datasets, experiments held within this second experimental study shed light on where EDL currently stands when it comes to the optimization of trainable parameters.

Table 2.9: Datasets used in Case of Study II.

Dataset	**Shape**	**# classes**	**# Instances (N_{train}/N_{test})**	**# trainable parameters**
HANDS	$30 \times 40 \times 1$	10	10,000 / 10,000	3,854
BCCD	$30 \times 40 \times 3$	2	17,000 / 5,416	9,065
MNIST	$28 \times 28 \times 1$	10	10,000 / 5,000	19,063
F-MNIST	$28 \times 28 \times 1$	10	10,000 / 5,000	36,188
GTSRB	$32 \times 32 \times 1$	43	20,000 / 10,000	83,999
CIFAR-10-G	$32 \times 32 \times 1$	10	10,000 / 5,000	1,658,570

For each of these datasets a fixed DL architecture is considered, featuring a realistic level of complexity (given by its number of trainable parameters), and rendering a good prediction performance when trained via gradient back-propagation (Table 2.10).

The layer type and structural hyper-parameters of every layer compounding such models are also specified in the table. Thus, the aim of this section

Table 2.10: Models utilized for the second case of study. $\mathtt{C2D}_{N,x\times y}$ denotes a convolutional layer with N filters of n rows and m cols; $\mathtt{D}_N$ represents a fully-connected (*dense*) layer with N output neurons; $\boxplus$ is a 2×2 max pooling layer; $\odot$ is a 2×2 average pooling layer; and $\mathtt{Drop}_p$ is a dropout layer with rate p. Layers enclosed within $(\cdot)^L$ are concatenated L times.

Dataset	**Network topology and structural hyper-parameters**
HANDS	$\mathtt{C2D}_{8,4\times4} - \boxplus - (\mathtt{C2D}_{16,2\times2} - \boxplus)^2 - \mathtt{D}_{20} - \mathtt{D}_{10}$
BCCD	$\mathtt{C2D}_{30,3\times3} - \boxplus - (\mathtt{C2D}_{16,3\times3} - \boxplus)^2 - \mathtt{D}_{16} - \mathtt{Drop}_{0.7} - \mathtt{D}_{1}$
MNIST	$\mathtt{C2D}_{28,3\times3} - \boxplus - \mathtt{C2D}_{14,3\times3} - \boxplus - \mathtt{C2D}_{7,2\times2} - \boxplus - \mathtt{D}_{128} - \mathtt{Drop}_{0.2} - \mathtt{D}_{80} - \mathtt{Drop}_{0.3} - \mathtt{D}_{10}$
F-MNIST	$\mathtt{C2D}_{64,4\times4} - \mathtt{Drop}_{0.25} - \odot - \mathtt{C2D}_{16,4\times4} - \mathtt{Drop}_{0.25} - \odot - \mathtt{Drop}_{0.15} - \mathtt{D}_{70} - \mathtt{D}_{10}$
GTSRB	$\mathtt{C2D}_{6,3\times3} - \odot - \mathtt{C2D}_{16,3\times3} - \odot - \mathtt{D}_{120} - \mathtt{D}_{84} - \mathtt{D}_{43}$
CIFAR-10-G	$\mathtt{C2D}_{32,3\times3} - \mathtt{Drop}_{0.1} - \mathtt{C2D}_{64,5\times5} - \mathtt{Drop}_{0.2} - \mathtt{D}_{128} - \mathtt{Drop}_{0.3} - \mathtt{D}_{10}$

is not to evolve very precise, state-of-the-art networks, but to assess the limitations faced by Evolutionary Algorithms when used for training these deep neural networks. Table 2.11 summarizes the training hyper-parameters utilized for all image classification tasks under study.

Table 2.11: Training hyper-parameters used in the experiments.

	Adam		SHADE-ILS			
Dataset	**Batch size**	**Learning rate**	**Population size**	**N$_{\text{eval}}$**	**Initializer**	**Epochs**
HANDS	128	0.01	10	200	Glorot	20
BCCD	64	0.02	10	200		
MNIST	512	0.01	10	200		
F-MNIST	512	0.01	10	200		
GTSRB	64	0.02	10	200		
CIFAR-10-G	256	0.01	10	200		

2.5.2.3. Addressing RQ1: On the Importance of the Layered Neural Structure in the Design of SHADE-ILS

Once the experimental setup has been described RQ1 is addressed, namely, a quantitative analysis of the impact and viability of exploiting the layered structure of DL models by SHADE-ILS, similarly to what gradient-based solvers do when back-propagating the gradients. To this end, a evaluation of the aforementioned scheduling strategies devised for SHADE-ILS over the datasets and DL models under consideration is performed, and its performance compared to that of a naive application of SHADE-ILS that does not take into account any structure of the problem.

Table 2.12 summarizes the results obtained in regards to RQ1. Specifically, the average loss and accuracy measured over train and test subsets are reported for every dataset and SHADE-ILS schedule. Values are averaged over 5 independent runs of every (dataset,schedule) combination. In addition, results corresponding to the Adam gradient-based solver are also included as a reference. The best results among those yielded by SHADE-ILS are highlighted in bold.

Table 2.12: Average accuracy/loss (over 5 independent runs) corresponding to different schedules of the SHADE-ILS algorithm over the datasets under consideration. Results corresponding to the Adam gradient-based solver are also included as a reference.

	Adam		FULL SHADE-ILS		DOWN SHADE-ILS		UP SHADE-ILS		A-DOWN SHADE-ILS		A-UP SHADE-ILS	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
	Acc Loss	**Acc Loss**	**Acc Loss**	**Acc Loss**	**Acc Loss**	**Acc Loss**	**Acc Loss**	**Acc Loss**	**Acc Loss**	**Acc Loss**	**Acc Loss**	**Acc Loss**
HANDS	0.9983	0.9932	**0.9708**	**0.9553**	0.7038	0.6910	0.7150	0.7072	0.6492	0.6470	0.5253	0.5189
	0.0082	0.0296	**0.1082**	**0.1589**	2.6608	2.6894	0.8194	0.8705	3.5419	3.5750	1.3179	1.3533
BCCD	0.9611	0.9815	**0.8558**	**0.8489**	0.7499	0.7400	0.7527	0.7439	0.7852	0.7768	0.8212	0.8087
	0.1314	0.0566	**0.3321**	**0.3399**	0.5306	0.5377	0.5227	0.5322	0.4861	0.4962	0.4149	0.4301
MNIST	0.9480	0.9534	0.9524	0.9342	0.9747	0.9504	0.9748	0.9490	0.9719	0.9452	**0.9772**	**0.9508**
	0.1672	0.1534	0.1590	0.2326	0.0851	0.1685	0.0856	0.1798	0.0938	0.1828	**0.0764**	**0.1627**
F-MNIST	0.9636	0.9672	0.9265	0.9196	0.9489	0.9384	0.9422	0.9311	**0.9561**	**0.9447**	0.9435	0.9336
	0.1190	0.1100	0.2539	0.2940	0.1773	0.2157	0.2077	0.2407	**0.1576**	**0.1966**	0.2012	0.2317
GTSRB	0.7437	0.7046	0.2329	0.2355	0.3636	0.3473	0.3596	0.3504	**0.3956**	**0.3868**	0.3204	0.3133
	0.6957	0.7807	3.0994	3.1196	2.4532	2.4931	2.4304	2.4672	**2.2916**	**2.3323**	2.6405	2.6622
CIFAR-10-G	0.8347	0.6542	0.2628	0.2602	0.3696	0.3612	0.3708	0.3642	**0.3806**	**0.3790**	0.3765	0.3681
	0.4519	1.2418	2.0952	2.1087	1.7878	1.8023	1.7737	1.7867	**1.7483**	**1.7619**	1.7612	1.7787

Several interesting observations can be made after inspecting the above table. To begin with, for relatively small-sized DL models (i.e. those for the HANDS and BCCD datasets), FULL-SHADE-ILS suffices for obtaining good scores, even superior than those rendered by its scheduled counterparts. This goes in line with the examination of the related literature (Section 2.3), in which many contributions deal with model training using naive bio-inspired solvers, without taking into account the structure of the network. The above results confirm that when the number of network parameters is low, a powerful optimization algorithm can effectively (albeit not efficiently) find optimal values for the task at hand.

However, when increasing the complexity of the network, the trend changes, and the exploitation of the layered structure of the DL model becomes essential to maintain a good performance. This is specially remarkable in the GTSRB and CIFAR-10-G datasets, in which the accuracy values of FULL-SHADE-ILS degrade severely with respect to those attained by A-DOWN-SHADE-ILS. For networks of moderate size (MNIST, F-MNIST), accuracy differences between the scheduled and non-scheduled versions of SHADE-ILS become neglibigle. Nevertheless, in terms of loss metric values the difference results to be larger, showing evidence that SHADE-ILS performs better in terms of optimized loss when endowed with the automated schedule mechanism.

When comparing the accuracy and loss values measured over train and test, a quick glimpse at the table confirms that in general, the trained models do not overfit excessively. Pausing briefly at the case of the MNIST dataset, arguably one of the most utilized databases in this field. If the average loss values achieved by the Adam optimizer (0.1672 over train, 0.1534 over test) are compared to those of A-UP-SHADE-ILS (0.0764 over train, 0.1627 in test), one can state that A-UP-SHADE-ILS achieves lower loss values than Adam, thereby concluding that this scheduled SHADE-ILS variant is a better optimization algorithm for model training than Adam. However, the final goal of predictive modeling is to provide models that *generalize nicely*, namely, models that perform as expected when predicting unseen data instances. When placing the attention in the losses measured over the test set, networks evolved by A-UP-SHADE-ILS for the MNIST dataset seems to overfit, thereby providing lower accuracy scores than expected. A similar conclusion can be drawn in other datasets (e.g. F-MNIST), yet at a lower extent than MNIST. This leads to an interesting insight on the influence of overfitting that will be elaborated in depth in Section 2.6.

To conclude the discussion on this first set of results it should be remarked that all SHADE-ILS variants achieve low scores when the complexity of the network is very high (models corresponding to the GTSRB and CIFAR-10-G datasets). In these cases the large gaps to the accuracy and loss scores of the network optimized by the Adam solver indicate without doubts that this meta-heuristic algorithm is of no practical use for this level of complexity. Given that SHADE-ILS is specially tailored to deal with high-dimensional optimization problems, it is fair to conclude that Evolutionary Computation and Swarm Intelligence methods are still far from being a realistic replacement for gradient-based solver. Instead, these empirical findings should drive the interest of the community towards hybridizing bio-inspired algorithms with gradient-based information and/or solvers.

2.5.2.4. Addressing RQ2: Comparing Bio-inspired Optimization Algorithms to Gradient-based Solvers

Experiments discussed in the previous subsection have concentrated on the performance comparison between different layer-wise optimization schedules of the SHADE-ILS solver. In the analysis of results shown in Table 2.12 is also highlighted the large gap between the gradient-based Adam solver and the best performing SHADE-ILS schedule, specially for DL models of moderate-to-high levels of complexity. Although this remark provides a partial answer to RQ2, this second part of the study delves into the convergence of the training process when undertaken via SHADE-ILS and Adam, aiming to discern the reasons for these identified performance gaps.

This being said, the focus is placed on the results corresponding to MNIST and F-MNIST, which are illustrative of the conclusions that can be drawn from the overall set of performed experiments. The dual plots in Figure 2.5.a to Figure 2.5.d depict the loss/accuracy convergence plots measured over train and test subsets corresponding to Adam and the scheduled A-UP-SHADE-ILS variant of the SHADE-ILS algorithm. Plotted lines depict the average loss/accuracy over epochs computed over 5 experiments, whereas the shaded overlay areas denote their standard deviation. Net loss values

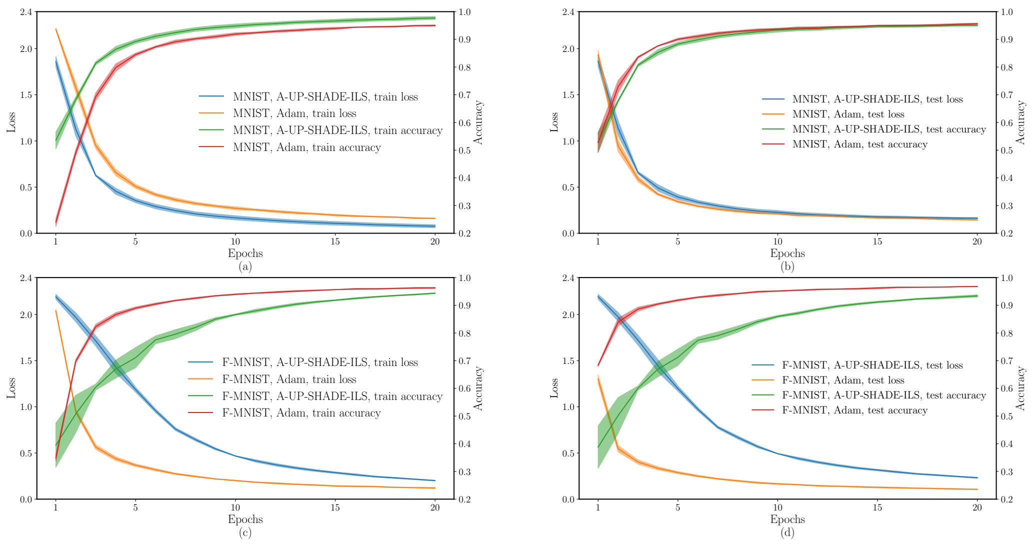

Figure 2.5: Accuracy and loss Convergence plots of Adam and A-UP-SHADE-ILS corresponding to (a) MNIST, measured over train set; (b) MNIST, measured over test set; (a) F-MNIST, measured over train set; (a) F-MNIST, measured over test set.

are indicated in the left axis of every plot, whereas accuracy score values are indicated in the right axis.

This discussion departs from Figure 2.5.a and Figure 2.5.b, corresponding to the convergence plots over the MNIST dataset over train and test subsets, respectively. It is straightforward to observe that when measured over the train dataset (Figure 2.5.a), both loss and accuracy scores of the A-UP-SHADE-ILS approach are better than those rendered by Adam over all epochs. This fact is conclusive in regards to the comparable or superior performance of SHADE-ILS when optimizing the trainable parameters of relatively small-sized networks. However, when shifting the focus on the test set (which reflects the generalization capability the evolved DL models), the curves plotted in Figure 2.5 reveal that A-UP-SHADE-ILS yields trainable parameter values that are slightly overfitted, thus generalizing worse than the model optimized via the Adam solver. This observation suggests that the apparently worse performance of Adam as an optimization algorithm is actually an advantage (a sort of *implicitly regularization* mechanism) that yields models of better generalization properties.

When increasing the complexity of the network to be evolved, Figure 2.5.c and Figure 2.5.d illustrate a rather different behavior of the convergence plots. At this point it should be recalled that the model selected to deal with the F-MNIST image classification problem has $36,188$ trainable parameters, almost twice the complexity of the model designed for the MNIST dataset ($19,063$ trainable parameters). The convergence curves in these plots show that although the accuracy and loss values of the networks evolved with both solvers get close to each other after all epochs are completed, A-UP-SHADE-ILS perform steadily worse than Adam over all intermediate epochs. In light of the results for the rest of datasets with models of larger complexity (Table 2.12), the case of the F-MNIST dataset must be understood as an inflection point, beyond which SHADE-ILS fails to perform competitively with respect to gradient-based methods. Furthermore, this worse performance occurs even if SHADE-ILS is allowed to execute more network evaluations per epoch.

These experiments and the conclusions drawn therefrom reinforce even further the initial belief that for the time being, current bio-inspired optimization algorithms do not constitute a feasible replacement for gradient-based solvers. In the next section the lessons learned through the literature analysis and experiments are collected and summarized, also several good practices and recommendations that should be followed to achieve significant advances in EDL are prescribed.

2.6. Advantages and Drawbacks of Evolutionary Deep Learning

As it follows from the experiments and the performed literature analysis, lights and shadows still remain in the application of Evolutionary Computation and Swarm Intelligence algorithms to the diverse optimization problems arising from DL.

2.6.1. General Lessons

The first lesson to summarize at this point is in close accordance with the general need for more methodological principles in meta-heuristic research across all the application scenarios where these solvers are applied nowadays. Unfortunately, the optimization problems tackled in this study are not an exception to this claim. The guidelines and procedures to be followed to reach solid and conclusive studies in the use of meta-heuristics are known to the community, specially in regards to the identification of the novel aspects of newly emerging algorithms and the proper design of comparison benchmarks. In this latter research stage, the assessment of the statistical significance is a must when dealing with problems related to DL topology and/or hyper-parameter optimization. Since several sources of uncertainty may coexist in the same problem statement (e.g. the operators of the search algorithm, the initialization of weights, and the stochasticity of the gradient-based training algorithm), reporting on the statistical significance via hypothesis testing should be considered a necessary step. Despite not an exclusive recommendation of the research area under study, leaving all code and results available in public software repositories for the research community is of utmost necessity, due to the dilated computation times usually required to run experiments with DL models.

Another general lessons that stems from the study carried out is that the goal in EDL is to yield models of improved generalization properties, namely, to find models that perform *better* when fed with unseen data. To an extent, in the conducted literature review it is noted that many contributions in the recent past dismisses this target goal and conclude that the solver at hand performs better since it achieves a more optimal objective value than gradient-based methods. Statements alike should be avoided in prospective studies, as there is no practical value in a model that generalizes worse *in the wild*, e.g. when predicting new data instances.

Another missing point in past contributions is a quantitative evaluation of the complexity of the solver, not only a mere indication of the quality of its produced output. As demonstrated through the cases of study, important complexity gaps exist between the solvers under study. Neglecting to inspect this important aspect of optimization solvers yields biased conclusions about the practical value of new proposals. Convergence plots like the ones depicted in the case of study II can provide a hint about the relationship between performance (accuracy) and complexity (number of epochs) of the solvers being compared. Along these plots, a clear definition of an *epoch* should be provided, so as to establish a reference and ensure fair complexity comparisons. Furthermore, for topological and structural hyper-parameter optimization the complexity of population-based meta-heuristic solvers is significantly higher than other schemes from the literature, specially gradient-based NAS search strategies such as DARTS [297], or RL-based NAS approaches [51]. Specifically, experiments with evolutionary NAS approaches such as those in [77, 89] require 3,150 GPU-days to complete their optimization task on the CIFAR-10 dataset, whereas DARTS only needs 4 GPU-days for the same purpose.

Definitely, more efforts are needed to validate whether bio-inspired meta-heuristics can reach the levels of computational efficiency of these alternative solvers or, at least, attain solutions of higher quality that make their computational overhead worthwhile. In this line, recent advances in the development of performance predictors of DL architectures [298, 299, 300, 301] can be an efficient workaround to the heavy computational requirements of model evaluation when population-based meta-heuristics are used for topological and/or hyper-parametric DL optimization.

Furthermore, other optimization objectives beyond predictive performance should also be considered in the future, such as the complexity of the optimized model in topology and/or structural hyper-parameter optimization. There are many reasons for considering such objectives in addition to predictive performance, ranging from an easier deployment of the optimized models in constrained computation hardware (e.g. cell phones) or a potentially more interpretable network structure [15]. However, most studies seem to focus just on the predictive performance of the optimized model, leaving unanswered relevant matters for model deployment such as the tradeoff between model's performance and complexity.

Finally, researchers should also work towards a global consensus on the dimensions of realistic benchmark dataset and models. A DL model is not just a layered structure of perceptrons, but rather a hierarchical composite of neural layers of different nature. It is their capability to extract increasingly specialized features from large-dimensional data what should bestow the label DL. Unfortunately, there are many works where DL refers to a neural network with very few trainable parameters and fed with already handcrafted features. The same can be said about the datasets used for validation, which in many studies lack the complexity that could argue the adoption of DL models. The community should agree on the minimum characteristics (task difficulty, diversity of datasets, model complexity) of experimental setups devised for reaching meaningful results and valuable conclusions.

2.6.2. Topology Optimization

Focusing now on the lessons learned from the literature that has so far tackled the optimization of the topology of DL models. First of all, the first case of study highlighted that the EvoDeep framework featured a diversity of layer types lower than that of AutoKeras, which restricts the search domain of the evolutionary algorithm that runs at its core. This fact can imprint a subtle yet impacting bias in prospective comparisons between new topology optimization frameworks. Such comparisons should ensure that the counterparts in the benchmark should utilize the same number and diversity of layer types when optimizing DL models for a given task. Otherwise, there is no certainty whether gaps found among the compared frameworks are due to the differences between their optimization algorithms or to the fact that some of them are in unfair disadvantage. Similar recommendations can be issued in regards to the formulation of the objective to be optimized, which should be set the same among frameworks.

In what refers to the optimization algorithm, it has been shown that bio-inspired solvers (in particular, evolutionary algorithms) are still far from the performance offered by other ad-hoc methods. Actually, the default optimization approach for the AutoKeras framework is a parameter-wise greedy strategy, and no Evolutionary Computation nor Swarm Intelligence methods are considered whatsoever. This observation implicitly suggests that bio-inspired solvers are still far from performing competitively, as the results in the first case of study have clearly shown. However, it is known that in other application domains, greedy search methods usually fall into local optima unless proper algorithmic countermeasures are included along the search. This opens up an opportunity to hybridize such greedy methods with global search heuristics or, alternatively, to design ad-hoc search algorithms that incorporate any of the ingredients featured by such greedy methods.

Another aspect in which much research effort has been invested is in the design of variable-size encoding strategies for the representation of evolved network architectures. This has been a subject of intense research for years since the advent of the first neuro-evolution frameworks, in particular the adaptation of compositional pattern producing networks made by Hyper-NEAT to represent and evolve neural networks. Despite the numerous works on neural representation strategies published thereafter to date, in many cases the selected encoding approach does not account for the validity of the sequence of neural layers it represents. To this end, EvoDeep resorts to finite state machines to model all possible transitions between layers, followed by a two-part encoding to represent global (training) hyper-parameters and layers' types and structural hyper-parameters, respectively. Another intelligent strategy is to search for optimal neural topology and structural hyper-parameter values over a continuous domain, allowing for the application of simple gradient-based solvers [302, 297]. All in all, it is coherent to belief that a key aspect for an efficient heuristic search is to tightly couple the solution encoding strategy to the design of the search strategy.

Finally, some words of reflection are dedicated to the level of granularity at which topology optimization should be performed. While in some recent works DL models are optimized at the level of the computation graph, namely, without assuming any particular type of neural layer. This extreme is interesting should the goal be to trascend conventional layer types and seek more diverse neural structures. At the other end of the scale, other contributions have capitalized on the mixture of pretrained modules, aiming to enhance the structure of the DL model while leveraging, at the same time, high-level knowledge acquired in other related tasks. To the knowledge if this study, there is no clear consensus on whether high-level or low-level topological optimization strategies are more promising for DL topology optimization.

2.6.3. Structural and Training Hyper-parameter Optimization

When it comes to the optimization of the hyper-parameters of the DL model, a first recommendation to elicit is to clearly specify the hyper-parameters to be optimized, as well as their search ranges. Echoing the aforementioned need for fairness in comparison benchmarks, and following the conclusions drawn from the Case of Study I (Section 2.5.1), it is crucial to guarantee that the compared solvers explore search spaces of equal complexity so as to remove any bias due to differences in this matter. A good practice for this purpose is to include a table listing each parameter with its corresponding search range, so that fairness can be guaranteed and reproducibility eased for the interested audience.

Another recommendation in hyper-parameter optimization is to estimate the impact of different hyper-parameter values in terms of the predictive accuracy and overall complexity of the optimized model. By reporting on the correspondence between different hyper-parameters values and the overall performance and complexity of the model, the community can discern potentially good search ranges for such hyper-parameters, thereby reducing the time needed to perform new hyper-parametric optimization tasks.

Finally, the taxonomy of the existing literature shown in Figure 2.3 revealed that the number of works simultaneously tackling structural and hyper-parameter optimization surpassed those dealing only with structural or training hyper-parameter optimization in isolation. This fact calls into question whether the results obtained so far for the latter cases are conclusive. The selected topology for the DL model affects directly the search space of a structural hyper-parameter optimization problem defined over it, so it remains uncertain whether the conclusions drawn for the particular network topology under choice can be extrapolated to any other network topology.

This is why it is suggested to increase the number of experiments with different network topologies and datasets when performing hyper-parameter optimization. Otherwise, the claims delivered by prospective studies can be in doubt due to the lack of enough empirical evidence.

2.6.4. Trainable Parameter Optimization

To end with, the optimization of the trainable parameters of DL models is arguably the one grasping most interest from the community in recent times. Several learned lessons and recommendations can be issued in this regard.

First of all, the research community should come to an agreement and understand that currently, EDL is far from fully replacing gradient-based solvers with bio-inspired optimization algorithms. The results discussed in the Case of Study II (Section 2.5.2) buttress this statement with solid findings: one of the most renowned and competitive algorithms for large-scale global optimization (SHADE-ILS) has not been able to perform better than the gradient-based Adam solver. Besides, when increasing the complexity of the DL model, the performance of SHADE-ILS degrades severely even if granted more computational budget (number of loss evaluations) than its gradient-based counterpart. In summary, for the time being bio-inspired optimization algorithms cannot rival the computational efficiency and the quality of solutions produced by gradient-based methods.

The main reason for this conclusion can arguably be found in the structure of the DL model, which establishes relationships among the trainable parameters of consecutive layers that *should* be exploited by the optimization algorithm. This is actually what gradient back-propagation realizes in a clever yet computationally efficient fashion, even though creating other known issues (e.g. gradient vanishing). In the experiments conducted it can be noticed that when adapting the search behavior of SHADE-ILS to the layered structure of the network, simple layer-wise schedules of this algorithm yields remarkable performance boosts, specially for networks of relatively small size. This suggests that more sophisticated hybridization strategies should be further investigated to embed problem-specific knowledge (namely, the structure of DL models or gradient information) within the search procedure of bio-inspired solvers.

Notwithstanding this noted performance gap, gradient-based solvers restrict the spectrum of loss functions to those for which derivatives can be computed. Furthermore, However, bio-inspired algorithms do not impose any requirement on the objective function to be optimized, nor do they require it to be differentiable. This latter fact could tilt the scale towards the use of bio-inspired algorithms in singular learning tasks that require a tailored definition of the objective function, as in cases with severe class imbalance or multilabel classification, or in network architectures with non-differentiable learning modules (namely, those through which gradients cannot be back-propagated).

When it comes to computational efficiency, the higher complexity of population-based meta-heuristics when compared to that of gradient-based solvers should stimulate more parallel and distributed implementations of Evolutionary Computation and Swarm Intelligence methods. There are modern programming languages and frameworks that can be utilized to accelerate bio-inspired search algorithms, even if still lagging behind the typical runtimes of gradient-based techniques. Recent works aim indeed at this direction, reviewing implementations available so far and prescribing recommendations and guidelines for the implementation of meta-heuristics in GPU [303, 304] and asynchronous distributed computing architectures [305]. Experiments with large datasets and realistic DL models should capitalize on already available software packages that ease the seamless deployment of meta-heuristics in massively parallel computing hardware, such as jMetalPy [306] (Apache Spark and Dask) and *libCudaOptimize* [307] (CUDA for GPU). Interestingly, Tensorflow (the computation engine that underlies well-known software libraries for DL models) also provides a naive implementation of Differential Evolution as one of its functionalities [308]. Parallel, federated or distributed computation frameworks for DL models are also spreading fast [86, 87, 309]. Definitely new studies should leverage the availability of these tools to undertake experiments at realistic complexity scales.

The learned lessons on trainable parameter optimization ends by emphasizing several good methodological practices that should be followed in prospective studies. First, the accuracy achieved by the optimized models over the test set should be informed jointly with the usual objective function statistics reported in experiments with bio-inspired meta-heuristics. This is particularly relevant in trainable parameter optimization to assess whether performance gaps identified between solvers do not come along with a penalty in the generalization of the evolved model. Furthermore, conventional gradient-based solvers should be always included in the benchmark, even if their lower complexity makes the comparison unfair in such terms. Finally, the use of convergence plots such as the ones depicted in Figure 2.5.a to Figure 2.5.b are recommended, as a visual tool to examine the relative differences between algorithms over epochs. This information can be very valuable for the deployment of the solvers in real hardware, as well as for the detection of overfitting issues like the one identified in the experiments conducted in this chapter.

2.7. Final Remarks and Opportunities

The exhaustive literature review and experiments performed have unveiled several promising facts and unsolved caveats of Evolutionary Computation and Swarm Intelligence algorithms when used for addressing optimization problems related to DL. Nonetheless, many proposals have been contributed to date for assorted learning tasks, not only supervised and unsupervised learning, but also other paradigms relying on DL models (e.g. deep RL).

Despite this noted activity, several research niches still remain uncharted or insufficiently addressed in this fusion of technologies.

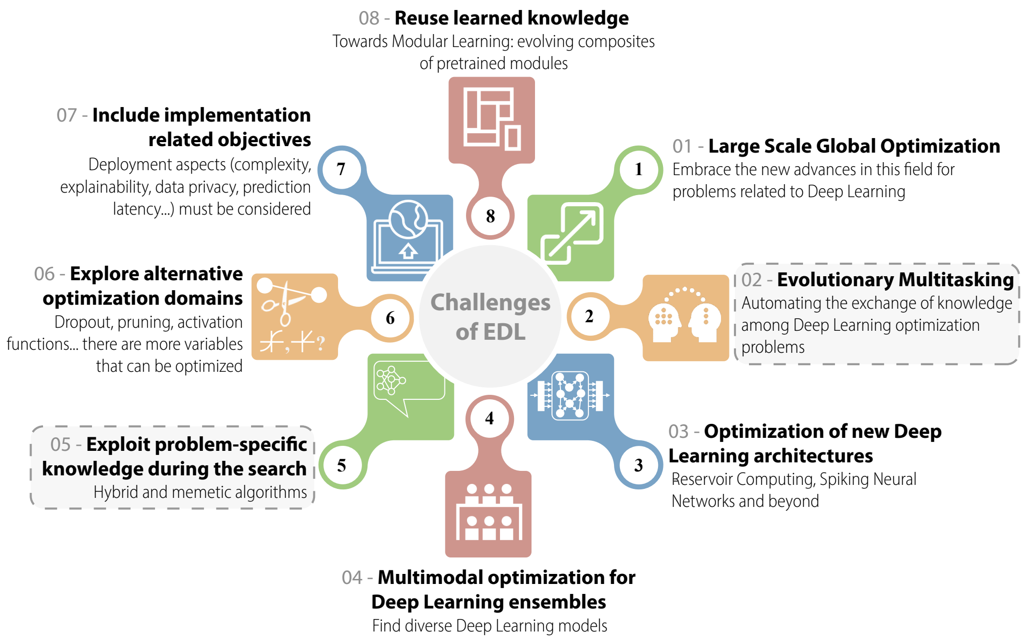
08 - Reuse learned knowledge
Towards Modular Learning: evolving composites of pretrained modules
07 - Include implementation related objectives
Deployment aspects (complexity, explainability, data privacy, prediction latency...) must be considered
01 - Large Scale Global Optimization
Embrace the new advances in this field for problems related to Deep Learning
06 - Explore alternative optimization domains
Dropout, pruning, activation functions... there are more variables that can be optimized
Challenges of EDL
02 - Evolutionary Multitasking
Automating the exchange of knowledge among Deep Learning optimization problems
05 - Exploit problem-specific knowledge during the search
Hybrid and memetic algorithms
03 - Optimization of new Deep Learning architectures
Reservoir Computing, Spiking Neural Networks and beyond
04 - Multimodal optimization for Deep Learning ensembles
Find diverse Deep Learning models
1
2
3
4
5
6
7
8

Figure 2.6: Challenges and research directions envisioned for Evolutionary Computation and Swarm Intelligence for the optimization of Deep Learning models. The challenges highlighted with a gray background box are covered in this Thesis.

In this section several challenges and research directions are summarized, which should be under the target of future efforts conducted in this area. Such challenges are schematically depicted in Figure 2.6, and contribute to the last two questions targeted by this chapter: what can be done in future investigations on the confluence between bio-inspired optimization and DL, and what future research efforts should be conducted for.

2.7.1. Large-Scale Optimization for Model Training

The design of bio-inspired algorithms capable of efficiently tackling large-scale optimization problems seems to be one of the critical points that require further developments to train DL models of realistic complexity levels. This is the reason for the selection of SHADE-ILS as the search algorithm in the second case of study. Indeed, SHADE-ILS remains nowadays as one of the most competitive proposals for large-scale global optimization, and is regularly considered as a baseline for competitions and benchmarks.

However, as in other research areas related to meta-heuristics, many advances in large-scale global optimization are regularly contributed to the community, featuring sophisticated ways to infer and exploit the correlation between variables during the search process (*interaction learning*). Improving this feature in large-scale solvers is often the main target of new proposals, either in an implicit fashion (as in Estimation of Distribution Algorithms, and Bayesian Optimization) or explicitly via grouping, statistical correlation-based methods, decomposition or other assorted means [310].

Unfortunately, the analysis conducted has revealed that most works related to trainable parameter optimization have resorted to off-the-shelf variants of bio-inspired solvers. Consequently, no consideration is made about the interactions between variables (weights, biases) that are known to occur due to the neural connections throughout the multiple neural layers. This motivates a closer look to be taken at new advances in large-scale global optimization, for both single- and multi-objective optimization problems [311]. Given the upsurge of DL problems in which more than one objective is established [78, 81, 80, 312, 313], the use of multi-objective solvers for large-scale optimization seems to be a natural choice.

2.7.2. Evolutionary Multitasking

An interesting research area has revolved lately around the design of evolutionary multitasking algorithms capable of simultaneously addressing several optimization problems within a unique search process that exploits the complementarities and synergies existing among such problems [17]. The challenge in this area is to develop intelligent optimization methods that not only promote the exchange of knowledge among candidate solutions corresponding to related problems, but also prevents the convergence of the search from being affected by *counterproductive* knowledge transfers among unrelated tasks.

The adoption of large-scale evolutionary multitasking to optimize simultaneously different DL models can boost even further the possibilities foreseen for the intersection between Transfer Learning and bio-inspired optimization. For instance, the transfer of pretrained modules between tasks can be conceived as a crossover strategy between networks partially evolved for undertaking different tasks. Similarly, the exchange of the parameters values between DL models can be also automated via evolutionary multitasking towards evolving behavioral policies for different reinforcement learning tasks [281]. Evolutionary multitasking has been also used to achieve modular network topologies [314]. The relative youth and promising results shown by evolutionary multitasking techniques are a sign of objective evidence that optimization problems related to DL should be explored via these techniques, e.g. by leveraging the straightforward exchange of knowledge among networks allowed by their hierarchically layered structure. Results later presented in this Thesis go in line with this observation, showcasing that this research direction has a long road of opportunities ahead. Furthermore, developments in multi-objective evolutionary multitasking [315, 316, 317] open up further opportunities towards considering other objectives beyond accuracy of relevance for DL, such as the complexity of evolved topologies.

2.7.3. Optimization of New Deep Learning Architectures

Most of the reviewed literature on bio-inspired algorithms for DL has focused on traditional forms of neural computation, including convolutional filters and recurrent units. However, this major activity has set apart the optimization of other neural network flavors, for which *Deep* (multi-layered) versions have been proposed over the years. Such alternative deep architectures have fewer optimization variables, hence favoring the use of naive bio-inspired solvers for the different problems that can be formulated on them.

One of such neural families is *Reservoir Computing*, which comprises a number of recurrent neural networks where only the parameters of the output layer (the readout layer) are learned. The parameters of the rest of recurrent neurons (the *reservoir*) are randomly initialized subject to some stability constraints, and kept fixed while the readout layer is trained [318]. Some works have been reported in the last couple of years dealing with the optimization of Reservoir Computing models, such as the composition of the reservoir, connectivity and hierarchical structure of Echo State Networks via Genetic Algorithms [319], or the structural hyper-parameter optimization of Liquid State Machines [320, 321] and Echo State Networks [322] using an adapted version of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) solver. The relatively recent advent of Deep versions of Reservoir Computing models [323] unfolds an interesting research playground over which to propose new bio-inspired solvers for topology and hyper-parameter optimization.

Despite more scarcely, optimization problems related other families of neural computation models have also been approached via Evolutionary Computation and Swarm Intelligence. The most remarkable case is the family of Spiking Neural Networks, in which topology and structural hyper-parameters have been addressed by means of different bio-inspired solvers [324, 325]. Training of synapses in spiking neural architectures has been also tackled in [326, 327]. The prospects outlined in the recent overview on training methods for Spiking Neural Networks [328] are also adopted for this study: efficient large-scale training methods should be investigated for new variants of these models, such as spiking deep belief networks and spiking convolutional neural networks.

2.7.4. Multimodal Optimization for Deep Learning Ensembles

Another interesting research path stems from the adoption of niching methods used in bio-inspired algorithms for multi-modal problems for the construction of DL ensembles (also referred to as *committees*. Indeed, the evolved population of candidate networks can be employed to retain near-optimal yet diverse DL model configurations. Such a diversity can emerge from different evolved topologies and/or values of their (hyper-)parameters. Such retained network configurations can be assembled into a committee, allowing for a robust fusion of their issued decisions.

Work in this direction has been published recently in [329], where a niching mechanism is used to penalize individuals that are more similar to others in the population are penalized. Network configurations remaining in the population after the search are then combined together via majority voting, showing a significant improvement in performance with respect to the use of a single evolved DL model. Multi-modal optimization methods can also be combined with diversity induction techniques (e.g. Novelty Search [330]) to promote an efficient exploration and discovery of multiple global optima over strongly multi-modal search spaces, just like the ones known to characterize DL optimization problems.

2.7.5. Exploitation of Problem-specific Knowledge During the Search

In light of the experiments, it can be deducted that the research community working on bio-inspired optimization should *go memetic* and exploit the rich structural properties of neural networks when designing new algorithmic approaches for any of the problems related to DL. A diversity of strategies can be followed for this purpose, from simplest layer-wise search schedules as the ones proposed in Case of Study I, to more sophisticated means like iterating between meta-heuristic and gradient-based search, or the use of similarity measures between neural layers [331] for controlling the behavior of search operators in topology and/or structural hyper-parameter optimization.

By blending together the global search capabilities of bio-inspired meta-heuristics and the problem-specific knowledge embedded in e.g. back-propagated gradients, a major performance improvement can be obtained over the optimization domains (structure, hyper-parameters, trainable parameters) under consideration, effectively overcoming known issues of gradient back-propagation approaches, such as exploding/vanishing gradients that lead to slow convergence. This is actually the approach followed in [49], yet validated with small network sizes. Another problem-specific aspect hybridized with bio-inspired algorithms can be found in [47]. In this work, the need for inducing curiosity in reinforcement learning models (specially in environments with sparse rewards) is realized by not driving the search of the learning agent with the reward objective, but rather with a measure of the *behavioral novelty* of the resulting policy. Other examples of hybrid methods are [332, 333, 244, 242].

This can be conceived as another example of the importance of considering the particularities of the learning problem in the design of bio-inspired algorithms. We firmly advocate for more proposals along this direction when addressing optimization problems in DL.

2.7.6. Exploration of Alternative Optimization Domains

The taxonomy around which the literature analysis has been organized considers three possible optimization domains on which a problem related to DL can be formulated: 1) topological variables, namely, the number and type of layers that compose the model; 2) hyper-parameters, which establish the details of the layers (structural hyper-parameters) and the optimization algorithm chosen for training the model (training hyper-parameters); and 3) trainable parameters (weights and biases). This threefold categorization collectively reflects most contributions reported to date in this research area. However, there are more optimization domains related to DL that can be tackled with bio-inspired solvers. One of them is pruning, e.g. the selective removal of connections among neurons for different purposes, from regularization against overfitting to a lower computational burden of gradient back-propagation solvers. Different pruning strategies can be developed to select which connections to drop at every layer of the DL model, which have been reviewed in recent comprehensive surveys on this topic [334, 335]. To the knowledge of this work, the use of Evolutionary Computation and Swarm Intelligence for neural pruning has been only done at the level of optimizing the dropout policy of the network [88]. When turning the focus on more fine-grained pruning strategies, large-scale optimization algorithms can be used to determine the subset of neural connections that must be discarded to achieve a good trade-off between generalization performance and network compactness. Recent findings in the application of genetic algorithms to convolutional channel selection [336] should be followed by further studies evaluating the scales at which network pruning with meta-heuristics can be realized. Besides pruning, other forms of network compression can surely benefit from the application of meta-heuristic algorithms, such as parameter quantization and sharing between layers or structures [337].

Other optimization domains for which Evolutionary Computation and Swarm Intelligence methods can be applied include the tailored design of activation functions [217], or the fusion of decisions issued within DL ensembles [127]. Definitely, these problems and other ones still to be proposed lay a magnificent panorama for bio-inspired optimization.

2.7.7. Inclusion of Multiple Implementation-related Objectives

When evolving DL models, objectives and constraints related to the application scenario on which they are to be deployed should be considered in the optimization problem. For instance, many software libraries and embedded electronic chips can be found nowadays in the market for the implementation and execution of neural network models in constrained computing devices, such as Internet of Things (IoT) sensors [338]. Likewise, real-world applications such as autonomous vehicular driving, remote precision surgery or wearable sensors restrict severely which DL models can be rolled out in their equipment. This suggests that a closer look should be taken at the complexity of evolved DL models during the optimization of their topology and hyper-parameters, among other domains.

A similar elaboration can be also made in regards to the progressive maturity of new paradigms such as Federated Learning [339] and Edge

Computing [340]. In these paradigms not only local models lack the amount of computational resources needed to run complex DL models, but also additional objectives are imposed. Efficient (incremental) training algorithms, energy consumed by the model [341], data privacy preservation [342, 343], robustness against adversarial attacks [344], or the explainability and accountability of decisions [15] are some illustrative examples of the eventual confluence of multiple implementation-related objectives in DL optimization. However, these factors are rarely taken into account in current approaches for evolving DL models. Most of them rely just on accuracy or any other measure of predictive performance.

This being said, DL models should be evolved by considering together several objectives and constraints as the ones exemplified above. To this end, it can be foreseen that bio-inspired algorithms for multi-objective optimization [345] can play a differential role in the future of DL. This branch of bio-inspired optimization, along with techniques devised to handle constraints during the search [346], can catalyze the practical deployment of DL models by addressing the improvement of the model's generalization performance together with implementation-related objectives and constraints.

2.7.8. Reuse of Learned Knowledge: Towards Modular Learning

The reutilization of pretrained modules between models corresponding to related learning tasks is at the core of Transfer Learning [16], whose most straightforward strategy is to reuse parts of a model developed for a task as the starting point for a model devised for another task. Such parts are often conceived as structural fragments of the network, particularly those capturing high-level features from the input to the DL model. Features corresponding to those parts are more easily reusable among tasks due to their high-level nature. In image classification, for instance, features learned by the first layers of the network are borders and broad shapes that could be of help for many different tasks. Therefore, it is intuitive to think that the output of such layers can be of help in other tasks for which annotated data instances are scarce.

The availability of DL models trained on different datasets and the effectiveness of just transferring layers between models corresponding to different tasks suggest a very interesting research path: expanding further the search space of topology and structural hyper-parameter search to also consider pretrained modules. The inclusion of such modules in the alphabet of possible layers could yield a major boost of DL models, specially for those cases with few labeled data. Furthermore, the flexibility of bio-inspired algorithms when designing the encoding strategy that represents networks during the search could allow achieving finer levels of granularity in the knowledge imported from such pretrained modules, to the scales of convolutional filters or recurrent units. A higher level of reuse can be achieved in the future with the consideration of meta-learning, namely, Several studies [316, 317, 318, 319] combine meta-learning and NAS to solve this problem.

There is a great opportunity to bring together transfer learning and topology/structural hyper-parameter optimization around the same goal: to evolve and discover DL models of superior performance.

2.8. Conclusions and Outlook

This chapter has presented a comprehensive and critical review on the use of Evolutionary Computation and Swarm Intelligence approaches to the topological, hyper-parametric and/or trainable parameter optimization of DL models. As previously indicated, the chapter is focused on three axes: a) definition of optimization problems in DL and taxonomy; b) a critical methodological analysis of the related literature and two cases of study, allowing to prescribe learned lessons and recommendations for good practices; and c) an enumeration of challenges and new directions of research. It is worth highlighting the aforementioned two cases of study, providing factual results on the performance of bio-inspired optimization algorithms when applied to the architectural design, hyper-parameter tuning and training of DL models.

The elaborations made throughout these three axes have yielded informed conclusions and insights about the four fundamental questions posed in the introduction, which round up the critical review of the field targeted in this chapter. The responses to such questions are synthesized below in the form of reflections stemming from the aforementioned axes:

1. **Why** are bio-inspired algorithms of interest for the optimization of DL models?

   The increased scales and diversity of neural layers of modern DL approaches have lately reactivated the global interest in DL optimization with bio-inspired algorithms, as a means to automate efficiently the processes of designing their topology, tuning their hyper-parameters and learning their parameters. Such processes can be formulated as complex optimization problems, motivating the adoption of bio-inspired algorithms for solving them efficiently. Furthermore, the renowned global search capability of Evolutionary Computation and Swarm Intelligence methods makes them a suitable choice to deal with complex search spaces as those characterizing DL problems. Finally, the flexibility of bio-inspired solvers to be hybridized with problem-specific search methods is another reason supporting the hypothesis that DL optimization can largely benefit from them.

   In conclusion, solid grounds for this synergistic fusion of technologies can be observed, which has so far stimulated the community to tackle optimization problems in DL using bio-inspired algorithms. However, it must be recognized that this fusion has not yet achieved results that are truly a step forward in terms of quality and objective achievement. This is still a lost race of bio-inspired optimization algorithms with respect to gradient-based solvers, which remain as the horse at the head of the race.

2. **How** should research studies falling in the intersection between bio-inspired optimization and DL be made?

   The discussion on the results obtained in the cases of study suggest that bio-inspired algorithms can be used for topological and/or hyper-parameter optimization, yet performing worse than other methods when comparisons are fair in terms of search space and complexity. Furthermore, the experiments have also revealed that even competitive bio-inspired solvers for large-scale global optimization are outperformed by conventional gradient-based solvers for trainable parameter optimization. These results, along with several issues detected in the literature (most notably, the unrealistic scales of the evolved model and the dataset/task under consideration), support the claims that there is a large space for improvement in this research area.

   On a prescriptive note, several learned lessons and recommendations have been identified, which trace *how* research should be done to reach solid conclusions and sound achievements. The most relevant ones are highlighted below:

   - Good methodological practices when designing experiments and benchmarks between different solvers, including realistic datasets and models, fairness in terms of computational complexity, assessment of the significance between performance gaps and reported performance scores over test instances, among others.
   - A closer attention at encoding strategies for topology optimization that account for the validity of the composition of layers that they represent.
   - A clear definition of the variable search ranges in structural and training hyper-parameter optimization, so that differences emerging between solvers can be attributed exclusively to their search efficiency.
   - The exploitation of problem-specific knowledge in trainable parameter optimization: the interactions between trainable parameters imposed by the hierarchical structure of neural connections should be exploited further by bio-inspired solvers for them to step out from the shadows of their application to the training of DL models.

3. **What** can be done in future investigations on this topic?

   In this regard, it has been underscored the need for overcoming the computational inefficiency observed in current bio-inspired optimization algorithms with respect to gradient-based solvers. It is known that these latter solvers have also their own drawbacks: they require differentiable loss formulations, they are sensitive to vanishing and exploding gradients and they are prone to local optima in non-convex search spaces. There are well-founded reasons why bio-inspired solvers can be a firm alternative to gradient-based methods, but more efficient designs and/or better performing implementations of bio-inspired approaches should be under active investigation in the future for them to become a practical choice for DL model training.

We have also stressed on other subareas in Evolutionary Computation and Swarm Intelligence of utmost interest for their application to DL optimization. Large-scale global optimization for training DL models of realistic complexity, or multi-modal optimization for the automated construction of DL ensembles, are just a few examples of the myriad of research niches in bio-inspired optimization that have not been explored yet. Evolutionary multitasking is another interesting direction to follow when approaching several DL optimization problems at the same time, mostly when such problems are related to each other and share a significant degree of overlap in their solutions (as occurs in transfer learning). Finally, optimization problems formulated on alternative DL models seem to have received less attention to date, and unleash further opportunities for bio-inspired optimization.

An additional research direction in this fusion of technologies has been identified in the existence of other variables and domains in which optimization problems can be formulated. Ensemble construction, network pruning or selective dropout strategies can also be described as optimization problems, favoring the adoption of bio-inspired optimization algorithms for their efficient solving.

4. **What** should future research efforts be conducted **for**?

   There is a strong incentive for which DL optimization should be approached via bio-inspired algorithms in the future: the consideration of additional objectives and constraints linked to the application scenario on which the model is to be deployed. Aspects such as the available computational resources, the need for periodically updating the model, or the time taken by the model for issuing a prediction should be considered during the design of the model for it to be of practical value.

   Furthermore, new paradigms such as Edge Computing, explainable Artificial Intelligence and Federated Learning have underpinned the need for taking into account other objectives beyond the accuracy of the model. Aspects such as data privacy preservation, the explainability of decisions issued by the model, the non-stationary nature of data at the edge, or the complexity of the learning algorithm can be formulated as additional objectives for the design of the model, impacting on its topology, hyper-parameter values and other optimization variables.

   These arguments, combined with the flexibility of bio-inspired algorithms to deal with multiple conflicting objectives, can be a primary *what for?* driver for adopting them to evolve DL models in practical settings. Specific scenarios and contexts that require ad-hoc designs to be evaluated with multiple objectives can and should also open the door to EDL models towards meeting imposed goals in terms of efficiency, performance and other application-related objectives.

All in all, there are promising evidences that Evolutionary Computation and Swarm Intelligence can tackle optimization problems related to DL, but they are not close to maturity yet, nor do they justify yet the replacement of other solvers used for the same purposes. Nevertheless, this is the role of research itself: to build upon the shadows of knowledge and bring light through scientific achievements. This work has just lit a candle to illuminate this path through the field of EDL.

Summarizing, this chapter has provided an in-depth overview of current trends in EDL, its pros and cons, and recommendations that can be drawn from the conducted literature study. In addition, two empirical cases of study have been designed and analyzed, whose results support the conclusions drawn throughout the review. In the next section, new trends in Transfer Optimization (TO) and Evolutionary Transfer Optimization (ETO) will be identified by following a similar literature study as the one performed in this chapter.

3 TRANSFER OPTIMIZATION AND EVOLUTIONARY MULTITASK OPTIMIZATION

"The right man in the wrong place can make all the difference in the world."
- Half-life 2

3.1. Introduction

Traditionally, optimization problems have been solved by different methods, part of which do not assume any a priori knowledge about the task under consideration. Over the years, this approach has demonstrated to be highly efficient in almost all real-world situations. Today, the scientific community has realized that this traditional way of solving problems may undergo some limitations. Indeed, the growing complexity of optimization problems and the fact that real-world optimization problems hardly appear in isolation have uncovered the need for exploiting knowledge gathered beforehand related to the problems themselves. This is the main reason why the incipient research area known as Transfer Optimization (TO [347]) has gained momentum within the Artificial Intelligence research community [17]. The fundamental aim of TO is to exploit the knowledge learned from the optimization of one problem (*task*) when addressing another related (or unrelated) problems, thus aligning much with the previously noted needs.

Up to now, three different conceptualizations of TO have been formulated in the literature. The first one, coined as *sequential transfer* [348], aims at solving problems that occur sequentially. To this end, the knowledge obtained when tackling preceding tasks is employed as external information when dealing with new problems/instances. The second one of these categories, referred to as *multitasking* [349], is devoted to the simultaneous development of different tasks by dynamically exploiting synergies existing among them. Finally, *multiform optimization* relates to the discovery of a solution for a single task found by using diverse alternative formulations.

Specifically, this chapter focuses on multitasking tackled through the perspective of Evolutionary Multitasking (EM) [350], also referred as Evolutionary Multitask Optimization. In short, EM seeks the development of efficient multitasking methods by relying on search procedures and operators drawn from Evolutionary Computation [351, 19] and Swarm Intelligence [352]. A significant effort has been conducted by the community for solving a wide variety of continuous, combinatorial, single-objective and multi-objective optimization problems through the perspective of EM [276, 353, 354, 315]. Another research direction for dealing with multitasking in the context of TO is multitask Bayesian optimization [355], which extends Bayesian optimization approaches to multitasking environments [356, 357, 358, 359]. Despite falling out of the focus of this chapter due to its non-evolutionary nature, it should be noted that Bayesian solvers, along those within EM, constitute the core of the contributions reported in the field of multitasking, with a significantly higher presence of EM methods.

A closer inspection at the most reputed scientific databases unveils that efforts carried out in EM are exponentially growing in recent times. This upsurge of activity demands a reference material to summarize achievements so far, detect and analyze research trends, perform a profound reflection on them to identify current limitations, and prescribe future research directions that push forward valuable advances in the field. This is the rationale for this chapter, which motivates its ultimate goal: to offer the readership of the Thesis a unified, self-contained and end-to-end outline of the activity reported around EM, towards properly understanding the methodological aspects that define this incipient field of knowledge, and how they have been utilized in the technical contributions of Part III of the Thesis.

Starting with, [360] exposes the work already done around the generic field of Evolutionary Transfer Optimization (ETO), providing an overview of existing studies gravitating on different topics related to ETO, namely, ETO for optimization in uncertain environments, ETO for multitask optimization, ETO for complex optimization, ETO for multi/many-objective optimization, and ETO for machine learning applications. The overview is supplemented by a set of challenges in the generic ETO research field. Having said this, this chapter takes a major step beyond [360] by elaborating on different directions that make this study differential on its own: a) a study fully focused on the stream known as Evolutionary Multitasking (ETO for Multitask Optimization in [360]), stressing on the algorithmic perspective, b) a manifold taxonomy based on three different pivotal axis: knowledge sharing pattern adopted (implicit or explicit), dynamic nature of the solving schemes (static or adaptive) and the design template of the search algorithm (MFO and Multipopulation-based Multitasking (MM)), c) a critical analysis of the methodological trends followed by researchers when designing and implementing EM-based methods; and d) an insightful discussion around challenges and opportunities fully focused on EM, in which topics ranging from possible applications, algorithmic enhancements and benchmarking issues are considered. Another review similar to the ones above can be found in [361], which presents a brief overview of the work done in the last five years in the field of multitask optimization and EM. Along with the definition and basic concepts of multitask optimization, authors provide a review of the field focusing on different concepts such as encoding schemes, parameterization strategies, knowledge sharing patterns, when and what to transfer, and evaluation and selection strategies. After that, additional aspects such as algorithm frameworks, many-tasking optimization problems or similarity measures among tasks are reviewed. Finally, authors analyze the main applications so far in the surveyed area, highlighting potential future works in the field. More recently, research around EM in the last years has been comprehensively examined in [362], orchestrating the literature review, current research trends and possible challenges around the main algorithmic components for designing EM methods.

	Definition & fundamentals	**Literature survey**	**Taxonomy**	**Methodological analysis**	**Critical analysis**	**Challenges**
[360]	✓	✓	×	×	×	×
[361]	✓	✓	×	×	×	✓
[362]	✓	✓	×	×	×	✓
This chapter	✓	✓	✓	✓	✓	✓

Table 3.1: Main contributions and differences among the literature study of this chapter and previously published surveys.

Despite the detailed dissection of the area offered therein, studies reported in [361] and [362] differ substantially from the one presented in this Thesis. The literature study departs from an alternative pivotal criterion and a taxonomy based on three axis that guides the critical discussion on EM. This taxonomy allows us to examine the work done in an intuitive and homogeneous fashion. Furthermore, the taxonomy-guided structure also facilitates the analysis of methodological trends followed by researchers, providing a valuable reference material for newcomers and early researchers arriving at the field. This chapter also includes a critical view of assumptions and core issues still unaddressed in EM. Lastly, a prospective on the most urgent and interesting challenges and opportunities in the area are offered, stimulating the reader to think beyond conventional paradigms in EM. Table 3.1 summarizes the similarities and differences among surveys published so far and the one presented in this Thesis.

The remainder of this chapter is structured as follows: Section 3.2 briefly poses the essential concepts of EM and introduces the reader to the main approaches used so far to face this paradigm. After that, Section 3.3 is dedicated to describe both followed bibliographic method and research questions that have guide the literature analysis. Next, Section 3.6 delves into the survey itself, departing from the presentation of the taxonomy criteria, to arrive at a careful examination of the recent bibliography related to EM. Equally important is the critical and methodological overview done in Section 3.4 and Section 3.5, bringing to the fore some critical aspects of the field and the main methodologies followed in the different phases of EM algorithmic development. Section 3.6 gravitates on the current limitations and discusses several challenges stemming therefrom. Finally, Section 3.7 concludes this chapter with a summary of the main conclusions and an outline towards the future of this field.

3.2. Definition and Essential Concepts

As introduced before, multitasking is devoted to the simultaneous solving of different optimization problems or tasks. It is important to emphasize at this point that the main goal of this paradigm is to find a promising solution to each of the problems at hand. This specific TO category is featured by an omni-directional knowledge sharing among tasks, potentially reaching a synergistic push between the problems being tackled [347]. In this way, multitask optimization sinks its roots in the premise that these complementarities among tasks lead to a competitive advantage over the case where the same problems are solved in isolation, either in terms of the optimality of the discovered solutions, or in terms of convergence and consumption of computational resources.

Mathematically, a multitask optimization scenario consists of K optimization tasks $\{T_k\}_{k=1}^{K}$, which are to be simultaneously solved. In this way, this environment can be characterized by the existence of as many search spaces $\boldsymbol{\mathcal{X}}^k$ as tasks. Furthermore, each k task has its own fitness function (objective) $f_k : \boldsymbol{\mathcal{X}}^k \to \mathbb{R}$, where $\boldsymbol{\mathcal{X}}^k$ is the search space over which $f_k(\cdot)$ is defined. Assuming that all problems should be maximized, the main objective of multitask optimization is to discover a set of solutions $\{\mathbf{x}_1^*, \ldots, \mathbf{x}_K^*\}$ such that $\mathbf{x}_k^* = \arg\max_{\mathbf{x}\in\boldsymbol{\mathcal{X}}^k} f_k(\mathbf{x})$.

Two main characteristics have stimulated researchers to deal with multitask optimization scenarios by means of evolutionary search operators. On the one hand, the intrinsic parallelism that brings a population of individuals which evolve together is well suited to deal with concurrent problems. In fact, several papers have already highlighted the benefits of this structure for dynamically unveiling synergistic relationships between tasks [17, 363]. On the other hand, the continuous exchange of genetic material along the evolutionary search allows all tasks to benefit from each other [364]. Considering the formulation introduced above, there are several ways for dealing with multitasking environments through the prism of EM, being two the most used approaches in the state of the art (depicted in Figure 3.1):

- The execution of a single search process over a unique population $\mathbf{P} = \{\mathbf{x}^p\}_{p=1}^{P}$ that contains the solutions to all problems, and that fosters the exchange of information among them through the application of crossover operators (as in e.g. Multi Factorial Optimization (MFO)). In this case, an aspect of paramount importance is that each solution $\mathbf{x}^p$ in the population should be evolved over an unified search space $\boldsymbol{\mathcal{X}}^U$. Thus, each independent search space $\boldsymbol{\mathcal{X}}^k$ belonging to task T_k can be translated to $\boldsymbol{\mathcal{X}}^U$ by means of an encoding/decoding function $\xi_k : \boldsymbol{\mathcal{X}}^k \mapsto \boldsymbol{\mathcal{X}}^U$. For this reason, each individual $\mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$ in $\mathbf{P}$ should be decoded to yield a task-specific solution $\mathbf{x}_k^p$ for each of the K tasks. In this context, the appropriate encoding strategy used for the individuals and the capability of the designed unified search space to represent all solutions $\forall\boldsymbol{\mathcal{X}}^k$ is crucial for an effective knowledge transfer between tasks. Specifically, the formulation of $\boldsymbol{\mathcal{X}}^U$ should be consistent with the level of overlapping among problems being solved.

- The deployment of several search processes that run in parallel, one for every task under consideration, which exchange information periodically as per a defined knowledge sharing policy (as in e.g. Multipopulation-based Multitasking (MM)). In this case, each search process operates on a task-specific population $\mathbf{P}_k = \{\mathbf{x}_k^p\}_{p=1}^{P_k}$, whose size P_k and search operators can be particular for task T_k and hence, differ from those used for other concurrent tasks. In accordance with previous notation, $\mathbf{x}_k^p \in \boldsymbol{\mathcal{X}}^k$ $\forall p \in \{1, \ldots, P_k\}$. In this case, the exchange of information is usually made in terms of solutions eventually exchanged between populations belonging to different tasks, so that a mapping function $\Gamma_{k,k'} : \boldsymbol{\mathcal{X}}^k \mapsto \boldsymbol{\mathcal{X}}^{k'}$ is needed to translate an individual $\mathbf{x}_k^p$ to the search space of task $T_{k'}$. This mapping function can be defined and particularized per every task pair or, instead, can rely on an intermediate unified search space, such that $\Gamma_{k,k'}(\mathbf{x}_k^p) = \xi_{k'}^{-1}(\xi_k(\mathbf{x}_k^p))$, with $\xi_k(\mathbf{x}_k^p) \in \boldsymbol{\mathcal{X}}^U$.

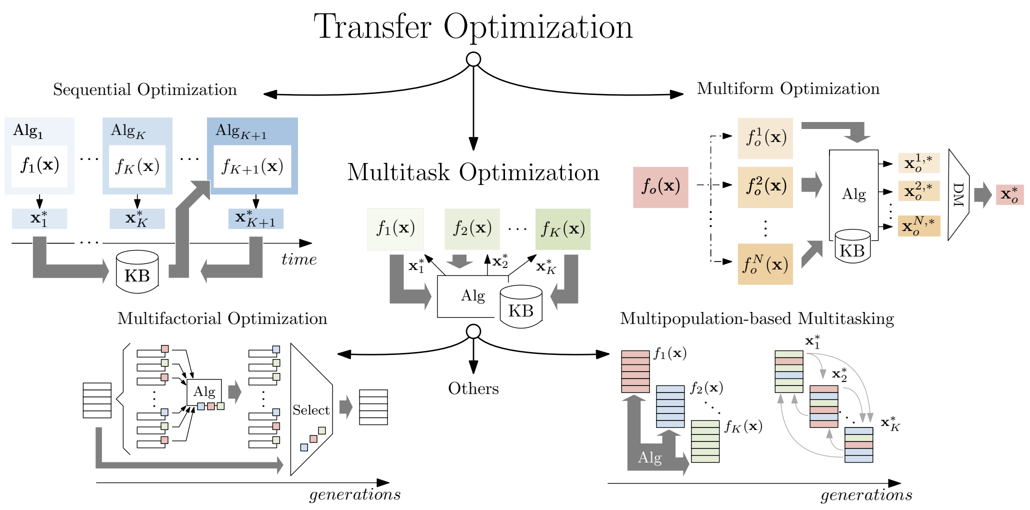

Figure 3.1: Schematic diagram showing the different ways Transfer Optimization can be realized, along with the two main family of algorithms by which multitask optimization can be approached using concepts from Evolutionary Computation.

Based on the work published by Ong and Gupta in [17], the overlap of two problems can be measured based on the amount of variables in the task-specific solution space which have the same phenotypical meaning. Thus, three different superposition levels can be identified depending on the amount of overlap in the phenotype space of the optimization tasks: 1) *complete overlap*, when tasks to solve are distinguished only on their task-specific auxiliary variables; 2) *partial overlap*, when problems share some characteristics, or tasks in which the distribution of variables is similar; and 3) *no overlap*, when problems to be tackled do not share any aspect of their structure. In any case, despite the relevance of the level of superposition when designing EM approaches, it is important to be aware that in many real applications it is not possible to measure the level of complementarity among tasks being solved without actually solving them [347].

Having introduced these concepts, it is appropriate to highlight that there is a common point of agreement in the related community, which states that EM was only materialized by means of MFO until late 2017 [365]. From that moment on, this incipient branch of TO has gathered a growing amount of contributions centered on the proposal of new EM solvers. Nowadays, it is widely agreed that two are the most recurring approaches for dealing with EM environments: MM and MFO, which conform to the two main design trends described above.

On one hand, MM approaches can be defined in a generalist way as techniques organized by different populations, in which each deme is devoted to the resolution of one specific task. MM can be heterogeneous, giving rise to different solving strategies relying on evolutionary and/or swarm intelligence heuristics or knowledge sharing protocols. Among these strategies, the one known as coevolutionary optimization (CoEV, [366]) is arguably the most frequently used today, in which knowledge sharing among populations (in terms of e.g., member migration or intra-deme crossovers) helps the evolution of each task. Examples of MM techniques are the multitasking multi-swarm optimization proposed in [367], the coevolutionary multitasking scheme introduced in [368] or the coevolutionary variable neighborhood search presented in [369].

On the other hand, the design of MFO techniques hinges on the definition of four different albeit interrelated specific concepts for each solution $\mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$ of the single population $\mathbf{P}$ over which the search is performed:

- *Concept 1 (Factorial Cost)*: the factorial cost $\Psi_k^p \in \mathbb{R}$ of an individual $\mathbf{x}^p \in \mathbf{P}$ is equal to its fitness value $f_k(\mathbf{x}_k^p)$ for a given task T_k, which can be computed after decoding $\mathbf{x}^p$ to $\mathbf{x}_k^p$ via $\xi_k(\cdot)$. Each member of the population has a list $\{\Psi_1^p, \Psi_2^p, \ldots, \Psi_K^p\}$ of factorial costs, each one associated with an optimization task T_k.

- *Concept 2 (Factorial Rank)*: the factorial rank $r_k^p \in \mathbb{N}$ of an individual $\mathbf{x}^p$ for task T_k is the position of this member within the whole population sorted in ascending order of Ψ_k^p. Every individual also counts with a factorial rank list $\{r_1^p, r_2^p, \ldots, r_K^p\}$.

- *Concept 3 (Scalar Fitness)*: the scalar fitness φ^p of $\mathbf{x}^p$ is computed based on the best factorial rank among the optimization tasks, i.e., $\varphi^p = 1/\min_{k\in\{1\ldots K\}} r_k^p$. This value is used for comparing individuals in a MFO algorithm.

- *Concept 4 (Skill Factor)*: denoted as τ^p, the skill factor is the task index in which member $\mathbf{x}^p$ performs best, that is $\tau^p = \arg\min_{k\in\{1,\ldots,K\}} r_k^p$.

The above four concepts are the cornerstone on which all MFO techniques rely. In fact, these definitions are used for different purposes, such as 1) deciding how population individuals interact with each other; 2) determining which solutions survive in the population between successive generations; 3) assigning tasks to individuals; or 4) classifying and sorting the whole population. With all this, these four concepts (either in their seminal form or in modified formulations, such as those proposed in [370, 371]) have led to several efficient MFO techniques for solving multitasking scenarios.

Furthermore, it is interesting to highlight here that two different knowledge sharing strategies can be found in EM methods, which can be approached as per the level of explicitness of the exchanged knowledge with respect to the evolved solutions. As such, *implicit transfer* refers to those cases where knowledge sharing is materialized through search operators, such as crossover functions. An example of implicit genetic transfer is the assortative mating used in most MFO techniques. By contrast, *explicit knowledge transfer* is conducted by migrating complete solutions from one task to another, which is often adopted in multipopulation schemes. Furthermore, it also be noted that explicit transfer could also be materialized through the use of mapping functions for transforming solutions before transferring, or by making use of Estimation of Distribution Algorithms (EDAs [372]). These alternative paths for knowledge transfer will be revisited when discussing the prospective on the field in Section 3.6.

All in all, a general pattern that reflect on the literature contributed to date can be discerned at this point. MM approaches lend naturally towards implicit knowledge transfer due to the direct exchange (*migration*) of individuals between subpopulations. Likewise, the unified search space of multifactorial optimization and the application of search operators over solutions encoded as such favors the interaction of tasks via mating strategies sensitive to the specialization (*skill factors*) of individuals. In other words, the design of the overall EM algorithm is tightly coupled with the explicitness of the knowledge transferred between tasks.

Notwithstanding the proven efficiency of EM solvers (including those related to MFO), it is appropriate to finish this section by underscoring that multitasking has been the focus of diverse debates questioning the efficiency of techniques proposed to date. Today, it is a clear consensus regarding the paramount relevance of the correlation among tasks to solve. The existence of these interrelationships is essential for positively capitalizing the shared knowledge over the search. Many studies have analyzed from different perspectives the similarities and possible synergies among problems [373]. However, in many practical environments it is not possible to quantify the existing complementarity among tasks in a preemptive fashion, without any knowledge of the optimal solution to each problem under consideration. This noted fact creates a latent problem for multitasking solvers, as the sharing of genetic material among non-related tasks is known to potentially lead to performance downturns. This phenomenon is known by the community as *negative transfer* [374], and has motivated a significant research upsurge towards alternative EM methods capable of avoiding and/or counteracting its effects in the convergence of the multitasking search. Such alternative methods will be reviewed and discussed in Section 3.6.

For the sake of understandability, it is convenient to clearly distinguish among several concepts that have been introduced in this section, and which are referred to throughout the rest of the survey: multi-population, co-evolution and distributed evolutionary algorithms. First, multi-population refers to all evolutionary algorithms whose population of individuals maintained over the search is structured into different groups or subpopulations. These subpopulations have their own algorithmic characteristics, such as different operators or parameters, allowing the search to be more diverse and rich. Multi-population may also include migration strategies, so that individuals can be shared between subpopulations under a certain criteria or migration policy. Co-evolution, however, implies the definition of two or more solving algorithms which communicate together so as to expedite the convergence of the search process. The communication among co-evolved algorithms is materialized through the sharing of valuable information obtained during the search. Finally, distributed evolutionary algorithms collectively denote those evolutionary solvers implemented in distributed processors. This parallelization can be implemented under two different strategies [375]: i) population-distributed models, which distribute an evolutionary task at operation, individual or population levels; and ii) dimension-distributed models, which focus on reducing the dimension of the search space tackled by every distributed solver.

Additionally, for the sake of a solid understanding of the EM paradigm, the following two sections clarify the main differences between multitask optimization, multi-objective optimization (Section 3.2.1) and multitask learning (Section 3.2.2).

3.2.1. Multi-objective Optimization Versus Multitask Optimization

An insightful reader can immediately relate EM to Multi-objective Optimization (MOO) paradigm which, when approached via evolutionary computation, span the wide family of multi-objective evolutionary algorithms. Indeed, it is possible to discern a conceptual overlap between both EM and MOO, since both aim at the optimization of a set of objective functions. However, as shown in Figure 3.2.a and Figure 3.2.b, these paradigms are completely separated from each other. On the one hand, EM aims to leverage the inherent parallelism enabled by a population of individuals for exploiting the synergies among related or unrelated tasks defined in different domains, each with its own solution space $\boldsymbol{\mathcal{X}}^k$ that potentially requires an encoding/decoding function for knowledge transfer. Moreover, EM also pursues the discovery of the best solution for every task. On the contrary, the goal of MOO is to find a set of solutions that differently balances between several conflicting objectives, defined over a single domain (and hence, over a single search space). In other words, MOO assumes the existence of a Pareto tradeoff between the objectives, for which the devised MOO algorithm produces an estimation in the form of a set of possible solutions. Therefore, there is no *unique* solution to each problem, but rather different solutions that meet every objective to a certain degree. In fact, EM setups where the tasks themselves are MOO problems can be found in the literature [315].

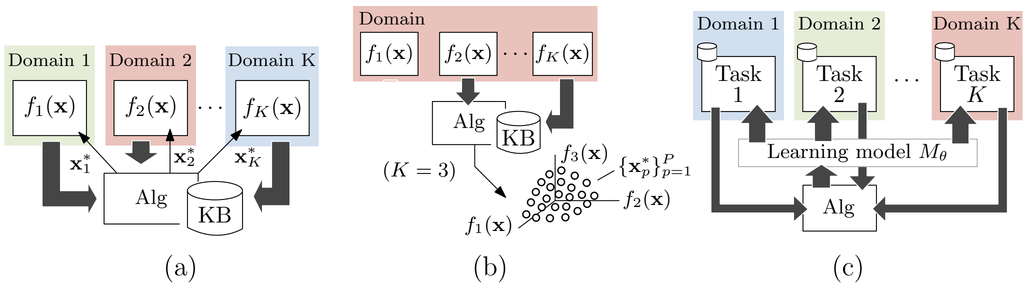

Figure 3.2: Diagram showcasing the core differences between (a) multitask optimization; (b) multi-objective optimization; and (c) multitask learning.

3.2.2. Multitask Learning Versus Multitask Optimization

Multitask learning and multitask optimization work on similar scenarios, in which a set of solutions $\{\mathbf{x}_1^*, \ldots, \mathbf{x}_K^*\}$ is sought for a set of tasks $\{T_k\}_{k=1}^K$. However, they mainly differ in terms of the optimization target, and the way in which the knowledge transfer is carried out. In multitask learning, the goal is to yield a model M_θ (with θ representing the parameters of the model) such that it can tackle the goal imposed by different tasks (e.g. classification of images of diverse kind). Here, the challenge is to determine a model structure and a value of their constituent parameters that best favors not only a good performance on every task under consideration, but also the exploitation of the synergies between modeling tasks. This dual functionality sought in multitask learning underlies beneath the design of multi-headed neural networks with shared losses trained via backpropagation, which are arguably the most utilized approach in the field: on one hand, sharing part of the neural architecture permits that part of the knowledge is common to all tasks, whereas the definition of a shared loss function ensures that the optimization of the parameters of the network is driven by the performance over all tasks.

This being said, there is a clear connection between multitask learning and multitask optimization, in the sense that multitask learning can be stated as a multitask optimization problem, provided that 1) solutions $\{\mathbf{x}_k^*\}_{k=1}^K$ elicited by multitask optimization represent the parameters of a model, and 2) solutions are constrained to part of their genotype being shared among tasks, so that they jointly embed a single model. This last constraint can be overridden so as to produce a set of models that collaborate together to solve several learning tasks more efficiently than in isolation. In this case, weeach optimization task would aim to seek the parameters of the model that best performs over the defined modeling problem for the task, and implicit/explicit knowledge transfer mechanisms used in EM could be effectively employed in place to transfer the knowledge learned in a certain task to another. All in all, multitask optimization must be conceived as a possible way of approaching multitask learning, but not the only one whatsoever.

3.3. Bibliographic Method and Guiding Research Questions

With the main intention of conducting a thorough and valuable survey, this section is dedicated to describe the research methodology followed for conducting this review. For properly gathering all the scientific material published around this incipient research field, several searches have been iteratively conducted using the most reputed and well-known scientific databases: the Clarivate Analytics Web of Science, Scopus and Google Scholar. At this moment, it is interesting to mention that due to the fact that Evolutionary Multitasking is a field that has not reached maturity, multiple terms have been employed for a systematic discovery of the published papers. The main reason for this situation is the non-existence of a general and well-accepted terminology for certain related concepts. For this reason, the next terms have been used for digging up all the produced material: "Transfer Optimization", "Multitasking", "Evolutionary Multitasking", "Evolutionary Multitask Optimization", "Multifactorial Optimization", "Multifactorial Evolutionary Algorithm" and "Multipopulation-based Multitasking".

Furthermore, after carrying out a first sweep using these terms, an exhaustive analysis has been performed paper by paper, in order to identify its adequacy for the present study. Once again, this situation is also a direct consequence of the youth of this field, which is why there are articles that using inaccurate terminology can lead to ambiguities. Having applied this second filter, and after the final selection of the articles to review, each manuscript has been categorized using the following criteria: knowledge transfer strategy employed, capacity of the solving approach for analyzing the negative knowledge and principal algorithmic scheme. Finally, and going deeper into the bibliographic analysis carried out, the three main research principles that have guided this investigation have been the following ones:

- To determine which are the predominant methodological patters that guide the current algorithmic developments, in terms of knowledge sharing among solving tasks, and adaptation to negative transfer phenomenon.
- To clearly identify which are the principal used mechanisms and operators for the evolution of Evolutionary Multitask Optimization Methods, both in MFEA and MM related schemes.
- To establish a strong basis for a prescription of methodological improvement areas and opportunities for future research, which are conducted in Section 3.5 and Section 3.6.

Embracing the method described in these paragraphs, the next section is devoted to presenting the taxonomy elaborated on the topic at hand, EM, and to outlining the main progresses conducted by researchers and practitioners to date.

3.3.1. Review of Current Evolutionary Multitasking Trends

As mentioned in the introduction of this chapter, the research activity produced around EM is growing at a remarkable path since the first formulation of this vibrant paradigm. The main objective of this section is to systematically review the most important works published to the date in the field of EM.

In order to appropriately guide this section, the taxonomy is presented in Figure 3.3 which covers all the studies contemplated in this review section. For organizing this taxonomy, and subsequently this section, a two-level approach is used to classify all the published material. First, the knowledge transfer strategy employed by the proposed solving method is considered (implicit or explicit). The second level regards to the capacity of the method to proactively analyze the negative knowledge sharing among tasks and dynamically react to this issue, seeking to reduce its impact in the algorithmic search. On the one hand, if the solving approach does not include any analyzing mechanism, it is considered *static*. On the other hand, an algorithm is considered as *adaptive* if it not only employs this kind of analyzing strategies but adapt its structure to the unveiled synergies among tasks (by modifying the parameters of the algorithm, for example). Lastly, once this categorization is conducted, articles are further classified taking as reference the algorithmic approaches used and proposed by researchers and practitioners. In this regard, *MFO based schemes*, *MM based approaches* and *other methods* have been considered.

With all this, this taxonomy sorts the literature according to these algorithmic schemes, being also valuable for distinguishing at a short glimpse those areas in which the community has so far place most of their attention. This literature overview, together with the methodological review conducted in Section 3.5, settle a stepping stone towards the critical discussion that will be held in Section 3.6 around the main limitations, opportunities and challenges that bring this area.

3.3.2. Theoretical Studies on Multitask Optimization

As mentioned, this whole section will revolve around the systematic overview of all the work done up to now on Evolutionary Multitask Optimization. This overview has been conducted though the perspective of both algorithm proposals and their knowledge sharing patterns. Nevertheless, it would be a big mistake if one leave aside the large number of paramount articles which have contributed in a crucial way to the establishing, advancement and understanding of this field. Specifically, these are theoretical papers, which address the knowledge area from a less applied point of view, in order to understand in an adequate way the ins and outs of the field.

These works are essential for establishing in the community the main pillars that make the research stream can advance in an orchestrated and efficient way.

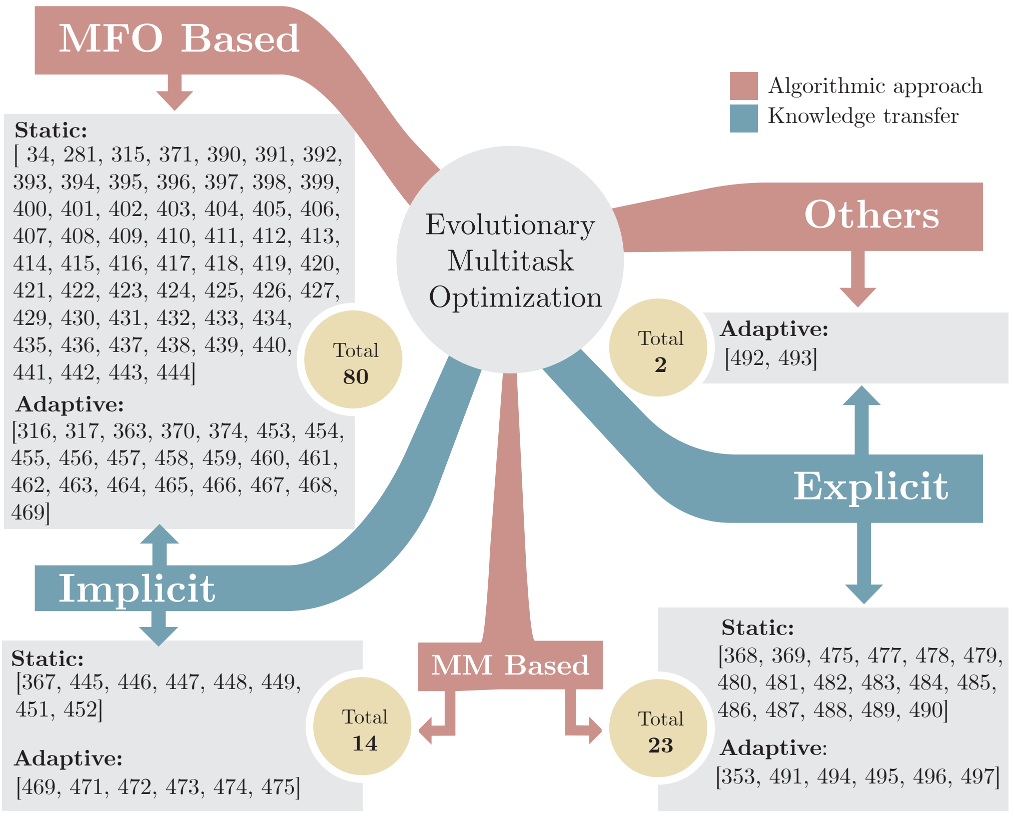

Figure 3.3: Taxonomy of the literature related to Evolutionary Multitask Optimization reviewed in this survey. A two-level classification has been made depending on the knowledge transfer scheme used and the adaptability of the proposed methods. Furthermore, solvers have been categorized into MFO and MM based ones, with an additional 'other' classification.

Probably, the most valuable paper in this context is this published by Ong et al. in [376], which is devoted to the introduction and presentation of EM field. This work is a cornerstone in the research community, establishing the basic concepts that have guided all the work conducted in last years. Apart from this influential and pioneering contribution, several remarkable theoretical works have been published on EM delving of different aspects such as the influence of complementarities between function landscapes on the search performance [377, 373], or just highlighting the main ingredients that make this knowledge stream interesting for the research community [349]. Further works on EM from a theoretical viewpoint can be found in [378, 17].

It is interesting to mention again at this point the study published by Gupta et al in [34]. That paper is not only significant for introducing to the community the most important method to date, MFEA, but also for establishing the principal wickers that make up MFO. As will be demonstrated in Section 3.3.3 and Section 3.3.4, both MFO and MFEA have been the source of inspiration for an abundant number of valuable works. As part of the work carried out, there are several published papers that have also delved into theoretical aspects of these paradigms. In the recent [379], for example, an analysis on the efficiency of MFEA is carried out. Main objectives of that study are twofold: to theoretically unveil why MFEA based methods perform better that classical techniques, and to provide some findings on the parameter setup of MFEA algorithm. In the recent [380], the impact of three different MFEA parameters is analyzed: probability of individual learning, probability of intra-crossover and probability of inter-crossover. In [381], a rigorous analysis is carried out on the relationship of MFEA and the conceptually similar multipopulation evolution models. To do that, authors make an in-depth comparison on their performance and working procedures. A similar study is also proposed in [382], revolving around the idea of the relationship among MFEA and island-based models. Interesting is also the brief study proposed in [383], focused on presenting insights in the measure of task relationship in MFEA. In [384], the efficiency of the binary tournament selection criteria used in the multi-objective variant of the MFEA is studied, proposing additional selection strategies. Further examples of theoretical studies can be found in [385], focused on analyzing the influence of the order of solution variables in multitasking environments; or in [386], devoted to the analysis of convergence in evolutionary multitasking scenarios compared to the conventional single task optimization.

Focusing on other theoretical aspects beyond those in MFEA, the position paper published by Gupta and Ong in [387] has become largely influential in the development of the field. The main objective of that research is to return to the roots of Evolutionary Computation. From here, authors provide an interesting review of the field for properly understand the inspirations of what can be classified under the umbrella of *multi-X evolutionary computation* concept. Thus, multitasking is again analyzed in this paper from its theoretical perspective. Also valuable is the recent work proposed in [388], in which a novel problem known as Competitive Multitasking Optimization is studied. In this new paradigm, solutions of solving tasks are compared to each other, so that its optimal solution is declared to the best among the optimal solutions of all the tasks under consideration.

Finally, it is interesting to mention in this category the studies described in both [389] and [365]. These reports contribute to the EM field by introducing some valuable test problems for both single-objective MFO and multi-objective MFO. Main intention of the authors of that works it to present to the community some heterogeneous benchmarks and baseline results, in order to use them for subsequent studies.

3.3.3. Implicit Knowledge Transfer Based Static Solvers

As mentioned in the previous Section 3.2, implicit transfer is often materialized through the application of dedicated search operators such as crossover functions. The principal standard-bearer for this type of transfer is known as *assortative mating* which is used in most of MFO techniques. The first paper revolving around *assortative mating* procedure is the same work which also introduces Multifactorial Optimization paradigm [34]. This paper became instantly in a reference paper, not only because of the introduction of MFO concept, but also for the formulation of the most used and influential EM technique: the Multi Factorial Evolutionary Algorithm (MFEA). From that moment on, many diverse adaptations and applications of the canonical MFEA has been proposed in the literature. In [390], for example, first discrete adaptation of MFEA is proposed, using as benchmarking problems four well-known permutation-based combinatorial optimization problems: Traveling Salesman Problem, Quadratic Assignment Problem, Job-Shop Scheduling Problem and Linear Ordering Problem. After that pioneering study, multiple additional discrete adaptations of the method have been proposed, such as the one focused on solving the Capacitated Vehicle Routing Problem in [391] or the series of works published principally by Thanh and Binh for the facing of clustered shortest path tree problems [392, 393, 394, 395, 396, 397, 398].

The interest in MFEA has stimulated several adaptations of this algorithm over the last years for efficiently dealing with real-world optimization problems. This is the case of the permutation-based MFEA proposed in [399] with cloud computing service composition purposes. A quite related approach was presented in [400], devoted in that case to the efficient semantic web service composition. In [401], authors develop a MFEA embedded with a greedy-based allocation operator for solving large-scale virtual machine placement problem in heterogeneous environment. An additional interesting application of MFEA has been recently proposed in [281], with the main goal of simultaneously evolving concurrent deep reinforcement learning models.

In addition to these adaptations, multiple advanced variants of the MFEA have emerged recently. These variants are mainly characterized by the adoption of novel mechanisms or operators, and they also rely on implicit transfer strategies for sharing genetic material between tasks. In [402], for example, authors introduced the named as Generalized MFEA. The main reason for the formulation of this technique is that MFEA experiences performance downturns when dealing with tasks with different dimensions, or problems whose optima do not lie in the same region of the solution space. The Generalized MFEA try to overcome these issues by implementing two different mechanisms related to decision variable translation and shuffling.

Another interesting variant of MFEA is developed in [403]. The improved MFEA proposed in this work explores the integration of a novel cross-task implicit transfer operator, which is based on a search direction instead of an individual. The main objective of this method is to accelerate the convergence of the search process, especially in environments where the optima of tasks are far from each other. Authors of [404] modeled an interesting hybrid MFEA which combines both MFEA and the Linkage Tree Genetic Algorithm. In [405], a variant of MFEA coined as *polygenic evolutionary algorithm* is designed, which curtails the cultural issues of the evolutionary procedure in the models of multifactorial inheritance. The main objective of that work is to understand the importance of both *assortative mating* and *vertical cultural transmission* towards effective evolutionary multitasking.

Further interesting MFEA variants have been proposed in [406] by means of the coined as (4+2) MFEA; in [407] introducing the MFEA with Individual Gradient mechanism for enhancing knowledge transfer; and [408] with MFEA with Priority-based Encoding. Furthermore, a variant coined as *potential individual-based MFEA* is proposed in [409] for solving the problem of constructing the data aggregation tree for minimizing the energy cost of data transmissions in a wireless sensor networks. A related problem is tackled in [410], in which MFEA with a network random-keys representation, a constraint-aware fitness function, and a novel crossover operator is proposed for solving the problem of prolonging the lifetime of wireless sensor networks. Further applications and variants of the MFEA can be found in [411] and [412].

Soon after using MFEA for single-objective optimization tasks, the research scope steered towards multi-objective optimization tasks, forging the so-called Multi-Objective MFEA (MO-MFEA) [315]. It should be noted here that this Multi-Objective MFEA also employs the *assortative mating* procedure for implicit knowledge sharing purposes. Furthermore, this specific method has been already used in a heterogeneous range of applications, such as for dealing with the multi-objective pollution-routing problem [413] or for the electric power dispatch [414]. A further application of MO-MFEA was presented in [415] for solving operational indices optimization. Improved variants of the referential MO-MFEA have been already proposed, such as the one in [416]. In that paper, authors introduce a MO-MFEA with a two-stage *assortative mating* method. This procedure introduced a preliminar division of the decision variables into diversity-related variables and convergence-related variables. After this first step, both types of variables undergo the *assortative mating*. Another adaptation was developed in [417], coined as decomposition-based MO-MFEA (MFEA/D-M2M). The main ingredient that characterizes this method is the adoption of a M2M approach for decomposing multi-objective optimization problems into multiple constrained sub-problems. The main goal of this procedure is to enhance the diversity of population and convergence of sub-regions. Also valuable is the study carried out in [418], focused on the resolution of the well-known multi-objective vehicle routing problem with time windows using an improved MO-MFEA by integrating bone route and large neighborhood local search. Furthermore, authors of [419] introduced a so-called Guided Differential Evolutionary (DE) MO-MFEA. Two are the main novel ingredients of this method: a) an improved crossover operator using guided differential evolution, and b) a modified Powell mechanism for mutation operations.

An additional improved version of the MO-MFEA can be found in [420], devoted to the solving of interval multi-objective optimization problems (cases in which the coefficients in their objectives or/and constraint(s) are intervals). The work in [421] also joins this profitable record of EM methods for multi-objective optimization tasks: in this recent work, a MO-MFEA method based on improved dynamical decomposition is presented. This approach, coined as MFEA/IDD, integrates the advantages of EM and decomposition-based evolutionary solvers. More concretely, in IDD a bi-pivot mechanism is implemented for providing an appropriate balance between convergence and diversity. Moreover, a MFEA-based method embedding the IDD mechanism is developed with the main objective of reducing the running time for solving diverse multi-objective optimization problems. Finally, authors in [422] present an enhanced version of the MO-MFEA, coined as IMO-MFEA. This method differs from the basic variant by including a novel mechanism for conducting more profitable crossover operators.

Quite differently to the aforementioned EM solvers, it is particularly interesting to pause at the single- and multi-objective optimization multifactorial evolutionary algorithm (S&M-MFEA) proposed in [423]. The main purpose of that solving scheme is to combine in a single multitasking environment the original single-objective MFEA formulation together with its associated multi-objective reformulation. Finally, in [424] authors proposed a MFEA with the incorporation of a prior-knowledge-based multiobjectivization via decomposition, with the main goal of building strongly related meme helper-tasks.

Despite the huge success and the contrasted efficiency of MFEA, researchers have rapidly detected the main limitations inherent to the canonical scheme of this method. As reported in previously published papers [374], the main limitation of MFEA is its difficulty for facing potential incompatibilities between different non-related tasks. For dealing with this issue, two principal research streams have been followed by the community up to now. The first one is the development of adaptive methods (as will be seen in Section 3.3.4 and Section 3.3.5). The other approach is the design of alternative solving schemes. Within this last category, different techniques can be found in the literature that address EM throughout the lenses of MFO but using a different scheme than MFEA. Another limitation of MFEA is that it resorts to non-structured populations, even though such a structure is assumed to exists. Thus, a more advanced and sophisticated structures could favor the design of alternative search operators, promoting a more controlled exchange of knowledge between related and non-related tasks.

Therefore, to overcome these limitations, practitioners have taken a step forward, proposing novel mechanisms which have led to the proposal of numerous methods, based on the essential concepts of the MFO. The first alternative MFEA scheme was proposed in [425], just some months later that the seminal work presenting the canonical MFEA. The main motivation that led the conduction of that work is to demonstrate that the practicality of population-based bi-level optimization could be enhanced by deeming the paradigm of EM within the search process. To do that, authors embedded the principal MFO concepts into the scheme of the well-known Nested Bi-Level Evolutionary Algorithm, giving rise to the coined as N-BLEA. Some months later, Sagarna and Yew-Soon introduced in [426] a MFO method for search-based software test data generation. In an attempt of leveraging the knowledge from different sources and enhance the search process, authors of that work proposed a MFO algorithm which bases the complete search procedure in mutation operations. Thus, authors evince that the selection operator and the preference relation used to compare individuals allow to inter-task knowledge transfer for an effective search.

Also proposed shortly after the introduction of MFEA, an interesting work delving in the main concepts of MFO can be found in [427]. The principal goal of that work is to explore the generality of the MFO paradigm, employing different population-based schemes. To do that, authors proposed the first multifactorial formulations of the hugely famous particle swarm optimization (PSO, [27]) and differential evolution (DE, [428]). Regarding the knowledge sharing strategies used in these DE- and PSO-based methods, they also employ the widely used concept of *assortative mating*, adapted to the mechanisms of the mmeta-heuristics at hand. Thus, they also rely on implicit transfer mechanisms. Indeed, that interesting work has served as guiding light for subsequent studies, such as the one conducted in [429], in which the performance of different mutation strategies in the knowledge transfer of multifactorial DE is studied. Furthermore, this same method is used as base in the remarkable investigation carried out in [430], which main goal is to identify the essential characteristics of tasks landscapes through the implementation of an inter-task evolutionary mechanism in the low-dimension subspace. Another example is the work proposed in [431], in which a MFO based PSO is proposed for feature selection purposes.

Another example of this scientific trend is the Multifactorial Cellular Genetic Algorithm (MFCGA, [371, 432]), which hybridizes the main concepts of MFO with the structural design and behavior patterns of well-known Cellular Genetic Algorithms. Main inspiration of that method is to have a more controlled implicit mating process among different tasks, favoring in this way the exploration and quantitative examination of synergies among the problems being solved. Also interesting is the approach introduced in [433] proposing a multifactorial particle swarm optimization - firefly algorithm hybrid technique. Main feature of this method is that individuals of the population can behave as a particle or a firefly, depending on the search performance. In any case, despite each member of the population can eventually move following each pattern, each individual maintains its nature along the complete execution. Further alternative MFO schemes can be found in [434], presenting a method for solving large-scale optimization problems called as evolutionary multitasking assisted random embedding; in [435], which introduces a MFO method hybridizing genetic transform and hyper-rectangle search strategies; in [436], which proposed an unified framework of evolutionary multitasking graph-based hyper-heuristic based on MFO concepts; in [437], which presents a MFO variant of the teaching-learning-based optimization algorithm; and in [438], which presents a random inactivation based batch many-task evolutionary algorithm, coined as IBMTEA-FCM. Additional MFO inspired techniques can be found in [439, 440, 441, 442].

Having analyzed alternative MFO schemes for dealing with single-objective optimization problems, it is worth mentioning further schemes devoted to solving multi-objective tasks. In this regard, the recent research conducted in [443] by Shen et al. can be highlighted, which introduces a novel multitasking multi-objective memetic algorithm for learning Fuzzy cognitive maps, inspired by the principal concepts of MFO. Furthermore, in [444] a novel multitasking method is proposed, which is fully devoted to the resolution of the sparse reconstruction problem. The method developed on that work was coined as multitasking sparse reconstruction (MTSR), and also relies of MFO concepts such as skill factor, factorial rank and scalar fitness for the multi-objective solving of the problem. It is also interesting to mention the genetic transfer scheme developed for that MTSR method, which is an enhanced variant of the *assortative mating* procedure coined as *Within-Task and Between-Task Genetic Transfer*.

The end of this section is dedicated to implicit knowledge transfer-based solvers by highlighting a few EM alternatives recently proposed which do not fully embrace the MFO paradigm. On the contrary, they adopt previously described MM schemes. A representative approach is presented in [367]. In that work, authors develop a dynamic multi-swarm method for EM. In that algorithm, the complete population is divided into as much swarms as task to solve. Furthermore, each subpopulation is divided into different sub-swarm. Thus, within each task subpopulation, a dynamic multi-swarm method is conducted. Furthermore, the knowledge sharing is realized through probabilistic crossover procedures with particles from other tasks groups, giving way to the coevolutionary factor of the method. Moreover, a parallel DE is proposed in [445], which introduces knowledge transfer patterns based on the archives of each DE solver. Interesting is also the MM technique developed in [446], focused on the multitasking adaptation of the well-known Fireworks Algorithm [447]; or the method based on genetic programming introduced in [448], in which tasks are solved in dedicated static subpopulations, allowing the crossing among individuals of different demes. Lastly, worth-mentioning is the multi-objective MM method proposed in [449], which adapts the well-known multi-objective optimization evolutionary algorithm based on decomposition (MOEA/D, [450]). In that algorithm, the implicit exchange of genetic material is produced through crossover procedures among individuals specialized on tackling different tasks. To do that, external neighborhoods are generated for each solving problems. Further methods of this category can be found in [451] and [452].

3.3.4. Implicit Knowledge Transfer Based Adaptive Solvers

As mentioned in Section 3.2 of this Thesis, a significant effort has been conducted by the community for overcoming the problems related to the so-called *negative transfer*. Examples of these alternative schemes are the adaptive EM methods. These instruments are mainly conceived for dynamically calculate the synergies among tasks, and subsequently measure how much knowledge should be transferred across different tasks. Thus, this section outlines those MFO methods proposed up to now to dynamically cope with the curse of *negative transfers*.

To start with, it is appropriate to mention the recently proposed MFEA-II [374], conceived as the evolved version of the standard-bearer method of the field: MFEA. Thus, two are the main ingredients embedded in the basic MFEA for evolving it to its adaptive variant MFEA-II. First, the parameter which dictates the extent of transfers (RMP) is now codified as a matrix, with a dedicated value for each pair of tasks. Second, this matrix is continuously adapted based on the performance of the multitasking search. It is also noteworthy that this method has been already adapted to discrete problems as can be seen in the recent work [453]. Furthermore, same authors that developed the single-objective MFEA-II introduced also its multi-objective version in [317]. As in the case of the static MFEA, these adaptive schemes also base their knowledge sharing on implicit procedures based on genetic crossover functions.

Another adaptive MFEA is proposed in [454]. In that paper, the method is endowed with a self-regulation mechanism. The main objective of this mechanism is to automatically capture the useful knowledge in common of the tasks at hand. For materializing this goal, this approach introduces the concept of ability vector, which substitutes the skill factor τ^p, and which reflects the solutions capability for tackling each of the optimizing tasks. Furthermore, similar authors that proposed MFEA-II in 2019, introduced two years before a Linearized Domain Adaptation MFEA (LDA-MFEA) [363]. This variant can be considered as an adaptive one, since it employs the linear transformation strategy for mapping the landscapes of a simpler tasks to the search space of complex ones. In that way, authors try to conduct efficient knowledge transfer between the problems while being optimized in concert. In the same year 2017, authors in [455] proposed a MFEA with parting ways detection and resource reallocation mechanisms. The first of this functionalities is in charge of detecting the occurrence of parting ways at which the sharing of knowledge is being unproductive, while the second mechanism reallocate fitness function evaluation on different types of generated solutions by ceasing the knowledge transfer when parting ways. Furthermore, in [456] a simple adaptive strategy for enhancing the resilience of the basic MFEA is presented. The method proposed therein employs a simulated binary crossover operator, followed by an adaptive mechanism based on information entropy for adapting the parameters controlling the MFEA evolutionary search.

Following this trend, a Group-Based MFEA (GMFEA) is modeled and implemented in [457]. GMFEA divides tasks into different conceptual groups depending on their proved synergy. Thus, GMFEA controls the implicit genetic transfer between problems belonging to same group. The most important feature is that the grouping is performed dynamically, without the requirement of any prior knowledge. Also remarkable is the research recently conducted in [458]. In that paper, authors first explore how diverse kind of crossovers impact on the implicit knowledge transfer in MFEA for solving continuous optimization problems. After that, they introduce a novel MFEA with adaptive knowledge transfer (MFEA-AKT), in which the mating function used for the genetic material sharing is autonomously adapted employing the information gathered on the complete search process. Furthermore, authors in [459] proposed a simple Self-Adaptive MFEA, which regulates the *rmp* parameter using a novel inter-task similarity measurement mechanism. Authors used their Self-Regulated MFEA for solving reservoir production optimization problems.

In addition to these single-objective MFEA variants, several works have been recently published focused on adaptively dealing with multi-objective optimization problems. Probably, the most remarkable work is [460], in which authors introduced a further adaptive variant of the MO-MFEA [460], giving rise to the canonical MO-MFEA-II. The specific method implemented in that work is characterized for introducing two novel ingredients: a) the deeming of a set of reference points to determine the diversity of current population (instead of using the crowding distance), and the online adaptation of the Random Mating Probability (RMP) with the intention of improving the genetic transfer of high-similar tasks. A further interesting method of this kind is developed in [316]. Specifically, this work is devoted to the implementation of a so-called MO-MFEA with decomposition and dynamic resource allocation strategy (MFEA/D-DRA). A further adaptive version of the MO-MFEA is proposed in [461], devoted in that case for the optimal operation of integrated energy systems.

Analogous to statirlc MFO algorithms, researchers and practitioner have also proposed several adaptive multifactorial methods inspired by the main concepts of this EM paradigm. As mentioned before, MFO methods are the main exponents of implicit knowledge transfer-based approaches. It should be highlighted first the adaptive multifactorial memetic algorithm proposed in [462], which congregates a) the use of local search mechanisms influenced by the knowledge learning among problems, b) a re-initialization procedure for overcoming premature convergence issues and c) a self-adaptive parent selection strategy based on search performance. Also valuable is the work conducted in [370], which is focused on developing an adaptive variant of the above mentioned MFCGA. The coined as Adaptive Transfer-guided MFCGA introduces two dynamic ingredients: a) a dynamic reorganization of cellular grids based on search performance and b) a self-adaptive multi-mutation mechanism. A further MFO adaptive variant can be found in [463], devoted to the presentation of a multifactorial PSO method with a self-adaption strategy for adjusting the inter-task learning probability. Similar authors proposed in [464] an additional adaptive method also based in the well-known PSO, following in this case the MM philosophy and implicit knowledge transfer pattern. Furthermore, in the recent [465], authors proposed an adaptive variant of the MFEA coined as Adaptive MFEA for RL (A-MFEA-RL) for simultaneously evolving multitasking reinforcement learning scenarios.

As multi-objective alternatives, can be highlighted the adaptive multi-objective and multifactorial DE algorithm (AdaMOMFDE) proposed in [466], based on multiple mutation operators which are selected following and adaptive strategy according to their search results. Also significant is the multiobjective and multifactorial subspace alignment and self-adaptive DE (MOMFEA-SADE) recently introduced in [467]. Principal ingredients of that method are a) a mapping matrix get by subspace learning and employed for modifying the search space and minimize the impact of negative transfers, and b) a self-adaptive trial vector used on the DE, for generating new solutions influenced by previous experiences. Also worth to highlight in this category the method proposed in [468] based on horizontal cultural-transmission mechanisms. Finally, the work conducted in [469] revolves around the multitasking adaptive formulation of the MOEA/D.

Finally, it should be highlighted that, also in this category, several MM solvers have been proposed in recent years in addition to those based on MFO. In this line, it is worth to describe first the coevolutionary multitasking framework proposed in [470], coined as evolution of biocoenosis through symbiosis (EBS). Inspired by the symbiosis in biocoenosis, EBS is comprised by multiple populations, running in each of them an independent Evolutionary Algorithm. Furthermore, the information exchange among tasks constitutes the so-called symbiosis, and it is conducted through an implicit transfer procedure coined as *Information Exchange through Concatenate Offspring*. Finally, this method introduces adaptive mechanisms for controlling information exchange, mainly based on the search performance. Further works on this method can be found in [471] and [472]. In [473], an adaptive solver based on genetic programming is proposed, devoted to the dealing of image feature learning. More specifically, this method generates different subpopulations in a dynamic way and the knowledge sharing happens through crossover procedures among individuals of different tasks. Lastly, [474] attempts at designing an efficient strategy for implicit knowledge transfer based on multidirectional prediction mechanism. The method proposed in this work divides the whole population into different classes based on binary clustering, calculating after that the representative point for each class. Once these points are calculated, the proposed algorithm generates multiple prediction directions by point, which are employed for generating new individuals through mutation strategies.

An additional remarkable co-evolutionary framework is proposed in [475], labeled as many-task evolutionary algorithm (MaTEA). This framework is similar to EBS in terms that it is also featured by having multiple populations governed by an Evolutionary Algorithm, each one dedicated to the optimizing of one tasks. Main characteristic of this MaTEA is an adaptive selection mechanism for choosing suitable assisted task for a given problem based on the accumulated rewards of positive knowledge sharing during the search. Moreover, a genetic material transfer schema via crossover is used for sharing information between problems for improving the efficiency of the search, giving rise to the coevolutionary nature of the method.

3.3.5. Explicit Knowledge Transfer Based Static Solvers

All the works mentioned in this chapter up to now clearly attest the importance that EM field has in the current scientific community. Furthermore, the intense activity highlighted in previous Section 3.3.3 and Section 3.3.4 also unveils the importance of MFO in this specific branch of kna travesowledge. In any case, this success cannot overshadow the fact that researchers and practitioners have proposed alternative schemes to MFO to deal with EM environments. Most of these schemes are MM based approaches, principally characterized by embracing explicit knowledge sharing strategies. This section is intended to outline the main work conducted in last years around explicit transfer based static solvers. As introduced in previous Section 3.2, this kind of knowledge transmission is usually conducted by migrating complete solutions among populations, namely from one task to another one. Additionally, explicit transfer could also be capitalized through the use of mapping functions or making use of EDA-style probabilistic models instead of raw solutions.

Arguably, the most successful alternative trend to MFO paradigm is the one related to MM approaches. Going deeper, most used MM methods fall inside the category known as coevolutionary. These multitasking methods are featured by being composed by multiple populations of individuals, which are usually independently dedicated to the optimization of a single tasks. Thus, the autonomous evolution of these subpopulations together with the punctual sharing of genetic material or the sporadic collaboration among them incurs in a better evolution of all of them in an unison way.

Some exponents of these methods can be found in the works [368], [369] and [476]. All these three algorithms are multipopulation approaches, governed by separated Genetic Algorithms, Variable Neighborhood Search and Bat Algorithms, respectively. These three methods have demonstrated a promising performance, using a scheme in which each subpopulation is devoted to the solving of one single task. Furthermore, the genetic material exchange is materialized through the punctual migration of complete solutions among the multiple populations. The same trend is also adopted in [477], in which a MM method named as Differential Evolutionary Multitask Optimization is proposed, in which the knowledge sharing is conducted through the migration of individuals among populations. A similar philosophy is followed in the Multitasking Genetic Algorithm modeled in [478], in which a population of solutions is created for each optimizing problem, and the knowledge sharing is realized at each iteration through the transference of different chromosomes among populations. The same paradigm is used in [479] for solving multi-criteria hyperspectral images band selection problems. More specifically, the subpopulations that compose the algorithm proposed in that work communicate among them for merging the bands information and accelerate the speed of searching promising bands.

Embracing the same research trend, in the research conducted in [480], an EM algorithm with explicit genetic transfer is presented. Also known as

EM via autoencoding, or Explicit EM Algorithm, this method is comprised by as much independent populations as task being optimized. The knowledge sharing is materialized along the search through the injection of good solutions found by any of the subpopulations along their execution. For appropriately conducting this genetic transfer, a multiplication operation is used with a previously learned task mapping. Authors of this last work extend their research in [481] by applying their EM via autoencoding to the well-known Capacitated Vehicle Routing Problem. Further evolution of this method is proposed in [482], with an algorithm coined as EMT/ET. That enhanced technique explores a novel selection of transferred solutions, based on the dominance of that solutions over the optimizing problems. Additionally, in [483] described a generalist multipopulation optimization scheme, based on similar concepts above described. Authors empirically demonstrate the efficiency of their scheme using a DE algorithm as base, giving rise to a so-called multipopulation multitask DE optimization. More concretely, this method capitalizes the sharing of information by sporadically creating overlapping populations.

Also interesting is the work proposed in [484]. In that work a specific instantiation of a multitasking genetic fuzzy system is presented and developed: a multitasking evolutionary optimization algorithm for Mamdani fuzzy systems with fully overlapping triangle membership functions (FOTMF-M-MTGFS). Further MM schemes can be found in [485, 486, 487], introducing surrogated-assisted mechanisms; in [488], which introduces a fast memetic algorithm; or in [489], in which a memetic MM algorithm is proposed for solving a capacitated vehicle routing problem.

Lastly, in [490] a multi-objective multifactorial immune algorithm is proposed. That MM method works with different subpopulations, and bases the knowledge transfer on an explicit mechanism coined as Dimensional Information of Solutions (DIS). Thanks to this mechanism, subpopulations exchange their individuals selecting tasks with similar iteration trends.

3.3.6. Explicit Knowledge Transfer Based Adaptive Solvers

To finish with this systematic review along the state of the art related to EM delving on the last category that can be found in the literature: explicit knowledge transfer based adaptive solvers. In this case, it is also interesting to mention that the methods than can be framed in this last category mainly embrace the above introduced MM philosophy.

In [491], an interesting adaptive version of the above described Explicit EM Algorithm [480] is proposed. Specifically, authors explore the use of the feedback gathered from the solutions transferred across tasks as guide for tasks selection. This feedback is updated along the search process, being able in this way to obtain the usefulness across tasks. An additional valuable algorithm is the novel EM algorithm with dynamic resource allocating strategy (MTO-DRA) introduced in [353]. The adaptive mechanism considered in this EM method is similar to those presented in [455] or [316]. Main novel ingredients of MTO-DRA in comparison with those similar methods is its multipopulation nature. More concretely, at each iteration, subpopulations are generated from the overall main population, each one fully devoted to the solving of one specific task. After this step, the resources are allocated to every subpopulation based on the index of improvements of tasks. This index is calculated online based on the performance feedback of previous generations.

Authors in [492] propose an online similarity learning strategy, named as adaptive model-based transfer (AMT). For demonstrating its good performance, authors instantiate an EM algorithm, called AMT-*enabled* EA. Main characteristic of the modeled AMT is its capability of dynamically learn and exploit the similarities across black-box optimization problems, minimizing negative transfers.

Interesting is also the recent work proposed by Lim et al. in [493]. In that paper, a multi-objective probabilistic model-based transfer evolutionary optimization technique is proposed, endowed with a solution representation learning mechanism. More concretely, aligned solution representations are learned through spatial transformations. Thus, the technique is capable of tackling handle mismatches in search space dimensionalities among solving tasks, and also of increasing the overlap between search distribution of tasks. It is also interesting to mention that algorithms proposed in both [492] and [493] count with a single population, not being classifiable as MFO nor MM. Additional alternative adaptive EM schemes can be found in [494, 495, 496, 497].

Throughout this systematic literature review section, a deep analysis on the efforts made so far in Evolutionary Multitask Optimization field has been conducted. In the next sections, some critical aspects of the field are further analyzed and discussed, also common methodological trends observed in the literature are considered. This critical and methodological overview should also serve as guidance for the upcoming challenges related to this promising field.

3.4. Critical Aspects of Evolutionary Multitask Optimization

Before proceeding with the methodological analysis of the field, there are several critical aspects that are often overseen in works related to the EM paradigm that should be spotted. This identification of unaddressed issues is supported by an exhaustive investigation of the existing literature, which has led us to the identification of several major concerns:

- To the knowledge of this study, no solid evidences have been given to support that the need for simultaneously optimizing several optimization problems is a task that occurs in practice over real-world scenarios. The immense majority of research advances on EM are validated over synthetically generated problem instances that are assumed to hold concurrently. Consequently, synergistic relationships between such problems can be modeled beforehand, laying a convenient yet not realistic playground for EM methods. Experiments over purely real-world problems are still in need for the field, informed together with a solid rationale of the plausibility of solving the problems involved in the experimentation at the same time with a single algorithmic approach. In short, is the concurrent appearance of optimization tasks a circumstance that occurs in real-world scenarios? What is a *synergy* between problems defined in the context of EM? Does the exploitation of such synergies depend on how they are defined? If so, can be estimated *in advance* whether such synergies exist, so that one can decide to solve them simultaneously? Recent efforts have been invested towards arguing that this research area has potential to crystallize in real-world applications [498], including the neuroevolution of robot controllers [374], unmanned aerial vehicle planning [407] or the optimization of last-mile logistics [481]. Nonetheless, an informed response to the above questions supported by real-world – rather than *realistic* – use cases is urgently needed.

- Some specific proposals – mainly those framed within the family of multi-population EM methods – can be reduced to extensions of classical well-known multi-population methods for single-objective optimization (e.g. co-evolutionary algorithms). Such extensions are endowed with subtle algorithmic adaptations to make them work within a multitasking scenario. However, in most cases the algorithmic novelty of such adaptations can be questioned, as they do not entail any innovative means to share knowledge between optimization. Instead, they resort to the same steps described when tackling single tasks, yet allocating dedicated resources (e.g., sub-populations) to every task under consideration. Another risk observed in the most recent contributions of the field is the use of biologically inspired metaphors to conceal the lack of algorithmic innovation of newly proposed EM methods. This aligns with the trend observed in other areas of bio-inspired computation [19]. The use of a well-established terminology to describe new EM proposals should be enforced to avoid ambiguities and findings of dubious scientific relevance.

- A growing number of studies focused on EM comprises biased experimental setups aimed at exclusively highlighting the benefits of knowledge transfer favored by EM methods. Researcher and practitioners working in this field should consider that the main challenge of the EM field is not only to improve the performance of multitasking methods in terms of exploiting commonalities between the problems in question. There is a further need for clearly verifying that EM methods are also better at solving the optimization problems at hand than using single-task competitive solvers and solving them in isolation from each other. Furthermore, most contributions related to EM neglect any prior knowledge about the problems to be solved, hence restricting the exchange of knowledge to that produced during the search. This assumption permits to gauge the exploitation of synergies that lies at the core of EM approaches, but is unrealistic and counterproductive for the search itself. In other words, the area may be losing sight on the final purpose of optimization research (*to efficiently solve a real-world problem*), emphasizing on EM elements (knowledge transfer) without assessing whether such ingredients provide a competitive gain over already existing solvers.

- In partial connection to the above, an interesting yet challenging possibility relates to the possibility of facing several optimization tasks of different nature by EM solvers, mixing together and exploiting synergies among, e.g. multi-objective, multimodal and dynamic optimization problems. Solving these heterogeneous tasks together could possibly increase the chances of discovering synergies among them, potentially leading to performance gains. However, studies addressing this research niche should depart from an informed conclusion about its plausibility and cope with the increased design complexity of the EM solver itself (e.g. appropriate solution encoding strategies that favor positive knowledge transfer across search spaces of very different nature). Most importantly, fair comparisons should be made against competition-winning solvers in each of the subareas of the tasks under consideration, so that the gains yielded by exploiting synergies among tasks are not outgained by solving them in isolation from each other (without exploiting any synergies), using optimization algorithms designed specifically for each problem type.

There is little doubt about the fresh breeze blown by EM over the evolutionary computation field. Progress on the field is being fueled by competitions organized at renowned conferences such as the IEEE Congress on Evolutionary Computation[1] or GECCO[2]. Although the number of contributions related to EM is rising dramatically as a result of this growing momentum, we definitely advocate for a close attention paid to the above crucial matters to ensure that EM research leads to knowledge that is valuable in practical settings.

Leaving aside this criticism and in accordance with the potential acknowledged for the field, in Section 3.5 the discussion is focused on the methodological trends identified after the inspection of the corpus of EM studies reviewed in Section 3.6.

3.5. Current Methodological Trends in Evolutionary Multitask Optimization

This section conducts a methodological overview of the current state of EM research field. The studies already published in this area have been really

1 IEEE WCCI 2022, accepted competition on Evolutionary Multitask Optimization: `https://wcci2022.org/accepted-competitions/`, accessed on February 11th, 2022.

2 GECCO 2020 Competition on Evolutionary Multitask Optimization, `http://www.bdsc.site/websites/MTO_competition_2020/MTO_Competition_GECCO_2020.html`, accessed on February 11th, 2022.

abundant up to date, giving rise to a significant amount of techniques which share common practices, mechanisms and resources. The main reason of the existence of these different research trends is because they are dedicated to tackle some latent implementation challenges that should be addressed when dealing with EM environments. Thus, the main goal of this section is to briefly highlight the principal methodologies adopted by practitioners in the different phases of algorithmic development. Such trends are summarized graphically in Figure 3.4 and next described in detail:

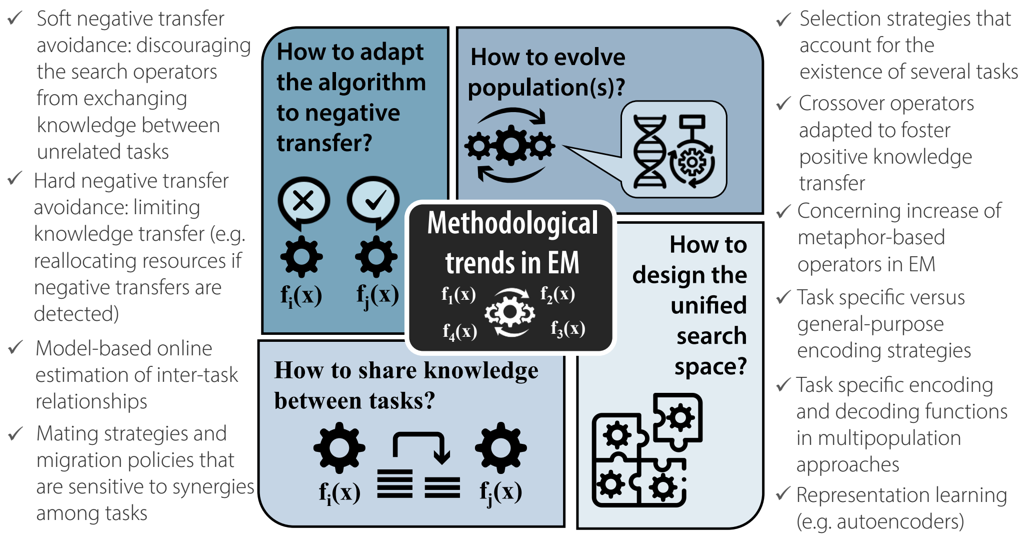
✓ Soft negative transfer avoidance: discouraging the search operators from exchanging knowledge between unrelated tasks
✓ Hard negative transfer avoidance: limiting knowledge transfer (e.g. reallocating resources if negative transfers are detected)
✓ Model-based online estimation of inter-task relationships
✓ Mating strategies and migration policies that are sensitive to synergies among tasks
How to adapt the algorithm to negative transfer?
$f_i(x)$
$f_j(x)$
How to evolve population(s)?
Methodological trends in EM
$f_1(x)$
$f_2(x)$
$f_4(x)$
$f_3(x)$
How to design the unified search space?
How to share knowledge between tasks?
$f_i(x)$
$f_j(x)$
✓ Selection strategies that account for the existence of several tasks
✓ Crossover operators adapted to foster positive knowledge transfer
✓ Concerning increase of metaphor-based operators in EM
✓ Task specific versus general-purpose encoding strategies
✓ Task specific encoding and decoding functions in multipopulation approaches
✓ Representation learning (e.g. autoencoders)

Figure 3.4: Main methodological trends identified in EM research as per the literature study presented in this Thesis.

- *How to design the unified search space*: One of the most important issues when facing EM environments is the way in which solutions are encoded. This is essential principally in approaches that fall inside MFO paradigm. The main challenge at this point is that wide and generalist encoding strategies will fall into superficial representations of solutions, not concrete enough for scrutinize interesting regions of task-specific search spaces. On the contrary, very specific representations can make impossible the genetic sharing between tasks coming from different optimization problems. In this sense, it should be taken into account that, depending on the type of EM technique to be implemented, it is possible that generated individuals are evaluated in different tasks throughout the whole search process. This is common in methods in which the sharing of knowledge is conducted by *explicit transfer*. In other cases, although individuals are dedicated to the solving of an exclusive task, the existence of *implicit transfer* procedures make essential that solutions devoted to the facing of different problems are capable of sharing knowledge with each other. This situation unveils the necessity of the existence of a unified search space, even more when tasks to solve are not completely related or belong to different typology of optimization problems. In the literature, many approaches for the efficient design of the unified search space can be found. If the tasks to solve are encoded by continuous variables, the most used method for encoding individuals is the well-known random-keys representation [499], as can be seen in works such as [315, 433, 454, 17, 463]. Furthermore, for discrete problems, two alternatives have been mainly followed by researchers: the transformation of the discrete search space to a continuous one through the random-keys representation, as mentioned in [34] and adopted in works as [391]; or the use of discrete search spaces such as the one introduced in [390] and used in works such as [370, 481]. Additional examples of encoding strategies can be found in the literature, but mainly constructed ad-hoc for a specific type of problems. Examples of this claim are the codification used in [392, 394, 395] for solving clustered shortest path tree problems; or the one employed in [369] for community detection over graphs. More recently, the use of neural auto-encoders has been proposed as a means to realize information transfer between tasks explicitly through the exchange of problem solutions, rather than delegating this exchange implicitly in the crossover operation over a unified search space [480, 481].

- *How to evolve the population(s) along the execution*: As mentioned before, Evolutionary Multitask Optimization refers to the design and implementation of multitasking solvers based on search procedures and operators drawn from Evolutionary Computation and Swarm Intelligence. Thus, as being population-based iterative methods, a crucial aspect that define this type of methods is the selection of the chromosomes that survive from one generation to another. In EM, several procedures have been proposed up to date, being the *scalar fitness based selection* of MFEA the most often employed one. Specifically, *scalar fitness based selection* is an elitist survivor function in which the best $\mathbf{P}$ individuals in terms of scalar fitness φ^p among those in the current population and the newly produced offspring survive for the next generation. Another alternative strategy is the coined as *local improvement selection*, by which the newly generated solutions can only substitute their direct parent if they improve it. This strategy is followed in methods such as MFCGA [371, 432], or those based on PSO or DE, as in [433, 354]. In another vein, in most of MM schemes, the survivor selection is conducted within each subpopulation, following traditional evolutionary computation or swarm intelligence selection operators.

Before proceeding further, at this point a pause is made to critically incide on certain practices that have been lately noted in this area: the derivation of new EM approaches in which the novelty exclusively resides in the use of new search operators, without any other research contribution to the EM area whatsoever. Despite the interest that the exploration of new meta-heuristic operators may awake in the community, this uprising corpus of literature should be appraised with care. As any other subarea of Evolutionary Computation and Swarm Intelligence, many voices are claiming to cease research efforts towards metaphor-based studies that lack any algorithmic novelty when compared to well-established meta-heuristic approaches [19]. Given its relative infancy, EM research should escape from these poor practices, and should focus strictly on algorithmic designs that connect closely to the core functionalities expected for an EM approach: new knowledge transfer mechanisms, unified solution encoding strategies, negative transfer avoidance over the search, and other parts alike.

- *How to share knowledge between tasks*: The effective genetic material transfer is arguably the most important factor for EM methods to work in an efficient way. This specific procedure is what makes a multitasking technique to be superior to classical solving mmeta-heuristics and schemes. In any case, the design of adequate knowledge sharing mechanisms is not trivial, and it usually depends on several issues, such as the encoding strategy employed, or the nature of the problems being solved. The main challenge on this point is twofold: i) to share as much as valuable knowledge among tasks and with an acceptable frequency, and ii) to define which is the genetic material that should be shared among optimizing tasks. Following these principles, the transference of knowledge can be capitalized following several directions. Probably the most used strategy, having shown great performance so far, is the generation of new individuals using genetic material coming from solutions with different skill task. Example of this specific trend is the well-known *assortative mating*, which can be materialized through i) a common crossover operation as in MFEA, MFEA-II and many other MFO techniques [34, 374, 370]; ii) based on mutation strategies as in DE inspired techniques [466, 429, 430]; or iii) the velocity based movements of PSO inspired methods [427, 433, 463]. Another commonly used mechanism for conducting intra-task knowledge transfer is the one used by MM methods, in which multiple populations coexists, each one devoted to the resolution of one specific task [368, 476, 478, 488, 369]. In that cases, the knowledge sharing is conducted mainly by migrating solutions among subpopulations, modifying in this way the optimizing task of individuals. Other less used genetic transfer scheme is the one based on a single-layer denoising autoencoder, used in the coined as EM via autoencoder [480, 481]; the generation of temporary overlapping populations [483], or based on the archives of DE solvers [445].

- *How to adapt the algorithm to negative transfers*: as has been seen in previous Section 3.3.4 and Section 3.3.5, a common trend for overcoming the curse of *negative transfer* is the design and implementation of adaptive mechanisms. The main motivation that inspires the development of this mechanism is also twofold: i) to share as much as valuable knowledge among synergistic tasks and ii) to avoid the inefficient transfer of genetic material among non-complementary tasks. In this regard, several promising alternatives have been proposed in the literature up to now. More concretely, two methodological trends can be distinguished: i) *soft negative transfer avoidance* mechanisms, which are devoted to discourage the knowledge sharing among non-related tasks, and ii) *hard negative transfer avoidance* mechanisms, aiming at prohibiting the transfer of genetic material among non-compatible tasks. Arguably, the most common used soft mechanism is the on-line fine tuning of algorithm parameters. This is the focal point of the model-based estimation of inter-task synergies embedded in the influential MFEA-II algorithm [374] and its discrete and multi-objective variants [453, 317], for example. Additional examples of this trend can be found in [460, 462, 463], using similar mechanisms as MFEA-II, or in [458, 370, 466], in which the adaptation is not in the parameters, but in the search operators used. Another common strategy is the resource allocation [455, 353, 316]. This mechanism is in charge of dynamically analyze the complementarities among tasks and allocating computational resources based on them. Another *soft* strategies contemplate the reinitialization of algorithm structures such as populations [370, 462] or the controlling of the amount of genetic material exchanged among solutions [470, 471]. As *hard* mechanism, it can be highlighted the dynamic generation of conceptual groups based on the arisen synergies [457, 475]. In any case, it should be highlighted that *hard* are more complex to implement, since they should be aware of the similarities among optimizing tasks (either in a preliminary way, obtaining them dynamically, or by studying the corresponding landscapes).

Despite these methodological trends, other great challenges and niches persist in the field. Some of these research opportunities are very closely linked to the trends discussed in this section, while others are devoted to finding new methodological approaches or the application of EM technique to new and more complex domains. These challenges are reviewed in Section 3.6, along with an outline of several research opportunities that are bound to attract much of the activity of the related community in the coming years.

3.6. Evolutionary Multitask Optimization: Challenges and Research Directions

Considering the review of the activity so far discussed in preceding sections, there is little doubt that Evolutionary Multitasking has brought a fresh breeze to the community working on Evolutionary Computation and Swarm Intelligence. Advances so far in this area have been notable, exposing the benefits of embracing multitasking in optimization problems close to reality. However, the relative youth of this field has left several challenges and research niches still insufficiently addressed. This section is devised to enumerate a series of open research questions, and propose some research paths that can be followed to tackle them effectively in years to come. Each identified challenge is complemented by a brief explanation of its scope, relevance and alignment with current research efforts made in other fields, summarizing all this information in Figure 3.5 for a quick visual reference of contents:

3.6.1. On Measures of Similarity Between Tasks: Are They Really Needed?

As discussed throughout the survey, so far the exchange of information between tasks has been done either implicitly or explicitly. In both cases, the similarity between tasks has been used to dictate which (and to a point, how) different individuals have been mated with each other, or to establish which tasks exchange explicit genotype information with each other. Notwithstanding this general usage pattern, an open question remains whether a priori assessment of the similarity between tasks is really needed in the context of Evolutionary Multitasking. Adaptive approaches such as MFEA-II- or ATMFCGA have exposed the capability of the evolutionary search process itself to elicit a progressively better estimation of the similarities between the problems being solved. However, there is no certainty whether this estimation of the similarity between tasks effectively avoids counter-synergies among them all along the process, particularly in early evolutionary stages. The availability of a priori information on how tasks relate to each other, by any means, should be exploited from the very beginning of the EM approach for the initial evolution to be informed properly.

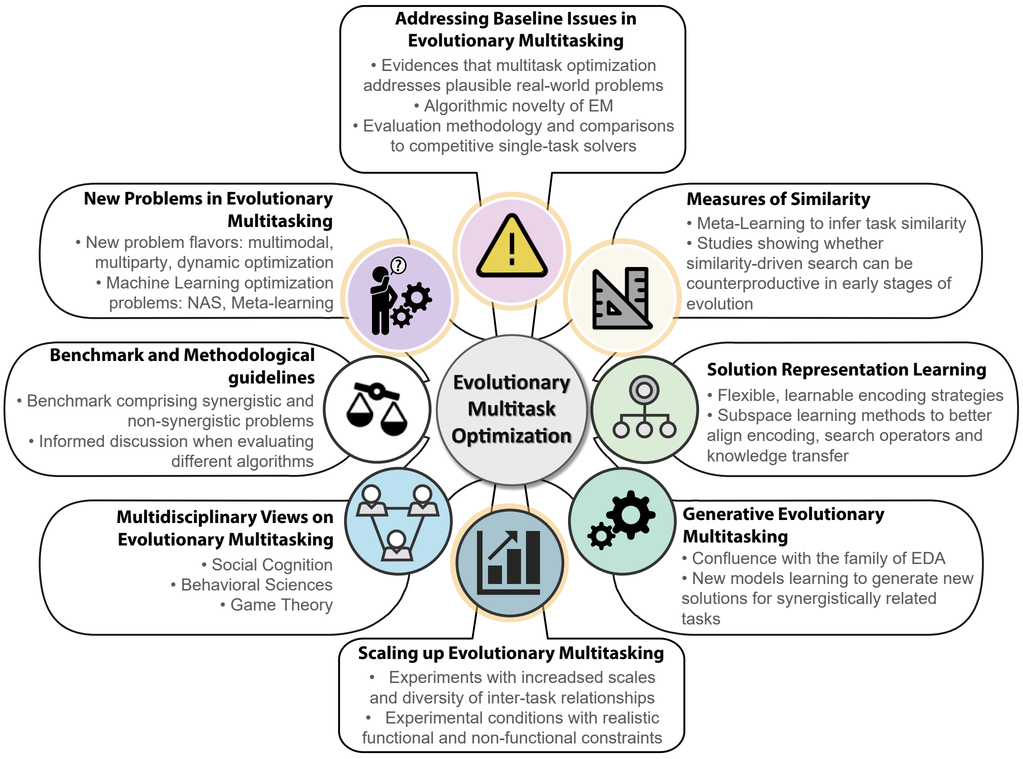
Addressing Baseline Issues in Evolutionary Multitasking
• Evidences that multitask optimization addresses plausible real-world problems
• Algorithmic novelty of EM
• Evaluation methodology and comparisons to competitive single-task solvers
New Problems in Evolutionary Multitasking
• New problem flavors: multimodal, multiparty, dynamic optimization
• Machine Learning optimization problems: NAS, Meta-learning
Measures of Similarity
• Meta-Learning to infer task similarity
• Studies showing whether similarity-driven search can be counterproductive in early stages of evolution
Benchmark and Methodological guidelines
• Benchmark comprising synergistic and non-synergistic problems
• Informed discussion when evaluating different algorithms
Evolutionary Multitask Optimization
Solution Representation Learning
• Flexible, learnable encoding strategies
• Subspace learning methods to better align encoding, search operators and knowledge transfer
Multidisciplinary Views on Evolutionary Multitasking
• Social Cognition
• Behavioral Sciences
• Game Theory
Generative Evolutionary Multitasking
• Confluence with the family of EDA
• New models learning to generate new solutions for synergistically related tasks
Scaling up Evolutionary Multitasking
• Experiments with increadsed scales and diversity of inter-task relationships
• Experimental conditions with realistic functional and non-functional constraints

Figure 3.5: Conceptual diagram summarizing the identified set of challenges and research directions in evolutionary multitask optimization, together with the baseline issues that still remain unaddressed to date. Challenges addressed in this Thesis have been highlighted in orange.

Departing from this last intuition, it can be foreseen that further efforts should be invested on advanced methods to estimate the similarity between optimization tasks without actually solving them. Clearly, a well-behaved measure of similarity between optimization tasks should roughly depend on the closeness of their optimal solutions. However, it is important to note that in the context of EM, the similarity between optimization tasks has no unique definition, and depends on the search and transfer operators being deployed. For instance, small differences in the solutions of two tasks can be amplified if the encoding strategy is not designed suitably, eventually leading to a counterproductive exchange of knowledge. This possibility is often overseen in the literature in favor of the design of unified representational strategies for all tasks under consideration. Conversely, given certain search operators, the tasks can be claimed to be related/similar to each other only in the context of the encoding strategy and operators in use, and provided that multitasking leads to faster convergence than in the case of isolated problem solving. The same set of problems may lead to negative knowledge transfer if different operators are used.

We definitely advocate for further research in this direction. A first research direction to follow is the incorporation of meta-learning algorithms capable of inferring the similarity between pairs of tasks based on meta-features extracted from the problems (e.g. based on fitness landscapes or on solution space sampling). This similarity estimation should be also complemented by an encoding alignment between tasks that ensures maximally aligned individuals in multi-population EM approaches. For this latter purpose, non-linear methods from domain adaptation have been recently explored from the transfer optimization perspective, leaving a door open to the consideration of further ingredients from subspace learning. In any case, models to represent and learn such synergies should account for the potentially fractional nature of synergies over the genotype space handled by EM methods, which should trigger major efforts towards synergy representation models that take into account this particularity.

3.6.2. Solution Representation Learning: Blending Together Encoding and Search Operators

Grounded on the two schools of thought about how to face information transfer between tasks (explicit versus implicit), a further step should be taken towards finding not only solutions to the problems, but also representation of the solutions for each problem that are more efficient for conveying knowledge transfer among tasks. This resonates with the reflections made in the previous subsection, by which similarity is strongly subject to the set of operators and the encoding strategy in use. Indeed, knowledge exchange between tasks can be beneficial only under appropriate solution representations. For instance, in graph coloring/community detection problems, a permutation-invariant encoding approach has been noted to be of utmost necessity for implicit information transfer through crossover [369]. Otherwise, information transfer cannot lead to better convergence, even if the networks to be clustered are rotated versions of the same network.

Solution representation learning is therefore vital for effective knowledge transfers. This is an exceedingly important topic for future research in evolutionary multitasking: learning solution representations, either based on prior data or adaptively during the course of the search, to enhance positive transfers. This unleashes an interesting opportunity for *learnable* encoding strategies, especially for those that can be evolved jointly with the solution itself (e.g. genetic programming). Otherwise, when allowed by the application domain where tasks are defined, tailored alignment methods or flexible encoding approaches should be utilized instead, always coupled tightly to the heuristic search operators in use.

3.6.3. Learning to Search using Generative Evolutionary Multitasking

Most EM approaches reported to date are based on sampling the space of possible solutions to the problem, without any attempt at learning the distribution of good solutions. In other words, the space of possible solutions is traversed by resorting to evolutionary and/or swarm operators, so that new solutions stem from the application of such operators to one or multiple populations of individuals. An additional degree of intelligence in how the space is sampled could be achieved by creating synthetic solutions along the search that reinforce and push forward the convergence of synergistically related tasks. If two tasks were found to be related to each other during the search, a generative machine learning model could progressively learn the distribution of *good* solutions for both problems. Once learned, this generative model could be queried over the search, replacing (fully or partly) the application of evolutionary operators. As a result, synthetic solutions that are potentially good for related tasks could be produced and fed to the population, ultimately accelerating the convergence.

The adoption of latent generative models already underlies beneath renowned EM methods, such as the probability mixture models used in MFEA-II to model the relationships between tasks. It is our belief that a profitable research path for EM remains in the long history of EDA algorithms, which address the concept of generative modeling and sampling for single-task optimization. It will be a matter of time when the EDA and EM realms collide together to span a new generation of intelligent multitask solvers, not only producing solutions, but also distributions that can be exported for other EM setups comprising task instances of similar kind.

3.6.4. Scaling Up Evolutionary Multitasking: Is It Just a Matter of Search Algorithms?

In reduced experimentation setups the use of EM methods has been shown to yield benefits in terms of convergence with respect to single-task optimization. However, when scaling up EM environments to realistic levels in terms of the number and diversity of tasks, these observed benefits can be turned down due to several reasons. To begin with, the computational resources required to scale up the search nicely with the number of tasks can become not affordable if the search over all tasks is to be made in a centralized fashion. An opportunity arises at this point for multipopulation schemes at the expense of multifactorial approaches, as they naively allow for decentralized implementations of the search and thereby, a more balanced share of the computational cost among stakeholders. This alternative, however, would come along with other aspects to be considered, such as the selection of a synchronous/asynchronous knowledge transfer policy or the reliability of the fitness evaluation made locally, among other issues noted in the field of distributed evolutionary computation [375].

Even if the above matters become eventually solved by the advance in research, definitely an additional question needs to be formulated: when and where can it be beneficial to solve thousands (potentially, millions) of tasks at once? Is there any realistic setup comprising these scales, in which several problems are related to each other so that this synergy leads to quantifiable performance gains? It is an undeniable truth that so far, experimental setups utilized in the community working in EM have been restricted to a few selected problem instances as per their relationship known beforehand. Non-functional aspects inherent to a distributed setup are, therefore, left aside in favor of a more focused pursuit towards algorithmic advantages. In real settings, however, other aspects should be under study, which could imply modifications at the core of EM approaches.

Among such aspects, promising paths have been lately traversed in what regards to the scalability of EM approaches with respect to the number and complexity of tasks [500]. In this work another important issue arising when deploying EM approaches in large-scale settings was identified: the efficiency of learning over the search how to transfer knowledge among a sparsity of related tasks. This work should stimulate increasing attention of the community in the need for a profound redesign of existing EM mechanisms when the practical setup in which they are evaluated comprises a more realistic mixture of related and unrelated tasks.

Privacy guarantees as those sought in affine modeling fields (e.g. federated learning, differential privacy, homomorphic computation) could be another aspect of relevance when scaling up EM frameworks. Delving into this matter, *federated optimization* would aim to evolve jointly different distributed tasks, without each task revealing each other their actual best solutions. A possible approach would be to define an encrypted unified search space, so that only each task could decipher its corresponding solution. The challenge in this direction is how to include this encryption functionality without jeopardizing the transfer of knowledge via implicit genetic transfer, or hindering the overall multitasking search efficiency.

3.6.5. Multidisciplinary Views on Evolutionary Multitasking: Social Cognition and Game Theory

Echoing the thoughts yielded from this Thesis about the use of EM in distributed environments, it can be concluded that there are other disciplines of knowledge from where the research community can find inspirations for new distributed mechanisms to share solutions to optimization problems. Among them, the interesting opportunities that can be identified in the field of social cognition should be highlighted, which in the field of psychology and behavioral sciences, refers to the study of mechanisms and procedures by which people process, store, and exploits information about other counterparts and the social contexts in which interactions with them are held [501]. In unregulated environments in which optimization tasks are addressed in a distributed fashion, elements from social cognition can be leveraged to dictate optimal ways to characterize the status of an agent (its search history), how to communicate it to other agents, when to do it, how to make an agent's search robust against deliberately negative knowledge transfer attacks. In essence, social cognition studies can give valuable hints towards establishing efficient, privacy-preserving protocols by which different solvers dialogue with each other to discover and eventually exploit positive commonalities. Concepts that closely relate to social cognition such as intersubjectivity, imitation, intentionality or confirmation bias can be extrapolated to the field of distributed multitask optimization, and can surely inspire new mechanisms to coordinate the transfer of knowledge among such tasks.

A similar line of reasoning can be followed in what refers to game theory. Game theory devises cooperative strategies among action and communicative tasks aimed to reach a mutual goal [502]. All in all, in EM the final goal is to achieve a good solution for all the optimization tasks under consideration. The essential role taken by the players' beliefs in game theory (which is bounded by the information it receives from other players and their observation of the consequences of their actions) connects this field to social cognition, and opens up new perspectives in the design of good strategies to endow optimization agents with the capability to reason about good solutions encountered by other agents during their search processes. Coalitions can be formed over the search, so that solvers that find their solutions to keep a degree of positive correlation over the variable and objective spaces can cooperate and coordinate the application of their search operators to traverse their solution spaces more effectively. The local exploration over the search space made by a solver should be influenced by prior results arising from the exploration of other agents belonging to the same coalition. This can be regarded as a similar relationship of interdependence than that between players in game theory, wherein every other players' possible decisions or strategies are taken into account when formulating a local strategy. Future efforts in EM should probe into this research niche, particularly in those scenarios where the tasks to be solved are distributed.

3.6.6. On the Need for Diverse Benchmarks and Methodological Experimentation Guidelines

One aspect of EM research that has been put to question is the quality of experimental benchmarks designed to assess the performance of every proposed approach. Most contributions to date have traditionally considered experimental setups comprising a few tasks, at their best belonging to 4-5 problem formulations over which inter-task similarities and synergies can be analyzed and discussed. Despite recent efforts in the heat of competitions held in frontline conferences[3], common methodological guidelines and benchmarking tools are still to be agreed and widely adopted in prospective contributions. Otherwise, there will be no clear grounds to ensure the fair evaluation, replication and comparison of new advances in the field.

Therefore, new benchmarks, quantitative metrics and methodological guidelines should be proposed, discussed and embraced by researchers working in EM. On the one hand, scores should relate to the quality of the produced solutions for the tasks under consideration, as well as the computational efficiency of the joint search, the gains with respect to single-task optimization, and the amount of positive/negative transfer episodes registered for every task pair. Finally, methodologically speaking all aspects that impact on the obtained simulation outcomes should be reported, especially those that are often overseen when describing the experimental setup (e.g. parameter tuning of all solvers under comparison, the imposed convergence criterion, and a solid justification why the selected parameters make the comparison fair among solvers). Furthermore, the usual discussion among approaches held on the basis of global performance statistics (e.g. average fitness per task) should be informed with additional null hypothesis tests [503] and/or a Bayesian characterization of the obtained results [287] to guarantee the statistical significance of the gaps claimed to exist among different approaches. All in all, a major push towards crystal clear comparisons in all measurable aspects of multitasking.

3.6.7. New Problems in Evolutionary Multitasking: Multimodality, Meta-learning and Beyond

In the last couple of years, a growing corpus of literature has addressed EM for tasks that go beyond real-valued single-objective optimization problems. This is the case of permutation-based combinatorial and multi-objective optimization, which have been tackled with EM approaches that incorporate algorithmic ingredients suited to deal with these problems. However, other flavors have been addressed more scarcely to date. This includes multimodal optimization, where synergies emerge as per the number and inter-distance between global optima shared by the tasks; dynamic optimization, partic-

3 Competitions on Evolutionary Multitask Optimization held at IEEE Congress on Evolutionary Computation (CEC'2017 to CEC'2021) and Genetic and Evolutionary Computation Conference (GECCO'2020).

ularly the case when changes undergone by two tasks occur in the same direction yet at different instants over time; or multiparty optimization, where several stakeholders participate, sharing part of the objectives and/or the solutions of their related Pareto front approximations.

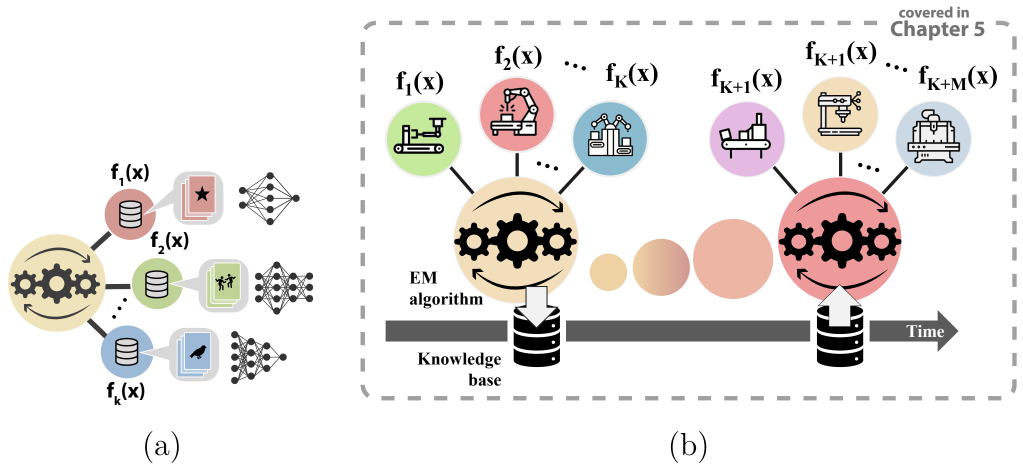

Figure 3.6: Conceptual diagram showing how EM can be used for (a) auto-ML, in which solutions represent the hyper-parameters of a ML model; (b) meta-reinforcement learning, where solutions represent policies of an agent when facing a given task.

An application domain that deserves a separate mention at this point is the confluence between Machine Learning (ML) and EM. Indeed, many learning algorithms can be formulated as an optimization problem (e.g. loss minimization in Deep Neural Networks), therefore unleashing an opportunity for undertaking setups consisting of several interrelated ML problems with EM approaches. For instance, it has been widely postulated that Evolutionary Computation and Swarm Intelligence solvers can be used as an scalable replacement for optimization problems related to Deep Neural Networks [504, 505, 506]. Initial explorations have exposed that EM can be used in multitask reinforcement learning environments to jointly train the neural models and exploit synergies between them [281]. However, other avenues at this crossroads are worth to be explored further, such as neural architecture search, where the joint evolution can serve as a mutual guidance for avoiding regions representing underperforming network configurations; and meta-learning, where the paradigm resides in how to optimize models that can perform well in unseen tasks (Figure 3.6). This is actually the main *leit-motiv* of the technical contributions of the Thesis explained in Chapter 4 and Chapter 5.

3.7. Concluding Remarks and Outlook

This chapter has elaborated on the research area known as Evolutionary Multitask Optimization. Framed within the wider Transfer Optimization field, the main goal of this incipient paradigm is to exploit the knowledge learned throughout the optimization of one problem towards addressing other related or unrelated problems, so that they are solved more effectively than in isolation. The relative youth of this area clashes with the high amount of contributions reported by the community in recent years. Consequently, this study aims at analyzing the past, present and future of this area, emphasizing on the methodological patterns, practices and biologically-inspired concepts followed by the community when designing new evolutionary approaches for multitask optimization.

To this end, the essentials of evolutionary multitasking are first visited, establishing mathematical grounds that allow discerning the aforementioned methodological patterns in subsequent discussions. Furthermore, a clear distinction between multitask optimization, multi-objective optimization and multitask learning has been done. Departing from these prior definitions, a systematic review of the literature related to EM has been performed, focusing on remarkable studies published in the last few years, and establishing a landmark taxonomy that allows the audience to easily understand the algorithmic choices mostly embraced in the reviewed bibliography. Specifically, this study has informed about the prominent role of multifactorial optimization and multipopulation multitasking approaches. Also, a methodological analysis of the different phases followed when designing EM solvers has been conducted, and briefly paused at several fundamental issues that remain without an informed answer: plausibility of the multitasking paradigm, overlapping algorithmic designs with solvers from other optimization areas, and comparison studies not answering the right questions. Finally, these thoughts have built upon the critical literature analysis conducted to initiate a discussion around research niches and challenges that remain insufficiently addressed to date. On a prescriptive note, each of such identified challenges has been associated with several possible research directions, which should inspire efforts in years to come.

The rest of the Thesis delves into the technical contributions carried out based on the opportunities identified in the literature reviews conducted in Chapter 2 and Chapter 3.

Part III

CONTRIBUTIONS

4

ADAPTIVE MULTIFACTORIAL EVOLUTIONARY OPTIMIZATION FOR MULTITASK REINFORCEMENT LEARNING

"In attempting the feat, one proves their courage."
- Hollow Knight

4.1. Introduction

Historically, EDL has shown to be a good substitute to traditional RL training techniques [46, 283]. A usual way to tackle the evolution of neural networks' structure and/or parameters is NeuroEvolution (NE [36, 118]). This research area takes advantage of EAs to work towards evolving diverse and well-performing neural networks. Although it has been applied to RL and video-game playing [61, 507, 116, 508], NE has also been used in other fields such as image classification [40, 509, 56], unveiling the potential of this kind of evolutionary approaches in the current Machine Learning context [510]. In fact, the spectrum of Evolutionary algorithms applied to DL is prominently expanding [280, 511, 257, 259], with knowledge transfer and multitask learning among the main challenges currently under the focus of EDL [504].

RL is known to require exploring huge search spaces in order to find good policies, capable of representing the behaviors required for solving a given task. For this reason, the knowledge acquired from previously trained problems is of utmost value for its exploitation in new tasks that share some amount of information. This exchange of knowledge is widely known as Transfer Learning (TL). In its seminal form, TL is realized by importing the parameters of a pretrained neural network for an untrained model devised for a task that share some similarities with the trained network domain. Although there is no holistic way to measure in advance the similarities between two tasks, transferring information between them has been proven to be beneficial for a myriad of learning problems.

Among such problems, this chapter focuses on multitask RL, in which the objective is to train a model or a set of models that can generalize to multiple RL tasks [512]. Hence, the design goal is to quickly adapt the model to previously unseen problem instances by taking advantage of the knowledge captured during the training phase. For instance, in multi-headed approaches [513] this information can be exchanged in the form of shared neural layers, or by means of distilled policies [514] as in multi-model approaches. Other strategies for promoting exploration in RL consist of the hybridization of the model with Evolutionary Computation, yielding the concept of Evolutionary RL. In this line of work, data diversification is generated in [515] by an evolutionary process, in which a model is partially learned using backpropagation, and partially evolved using the information of the gradient generated in the training phase. In another related contribution [257], diversity and exploration is enhanced by collaboratively evolving a set of agents using Evolutionary Computation, so that collaborative agents are able to perform tasks that failed to perform individually. EAs also provide many opportunities to promote the diversity of solutions. An example can be found in [282], where RL maze navigation task is evolved via DE while taking advantage of the diversity generated by the novelty search mechanism to avoid stagnation. Similarly, novelty search has been applied in [516] to increase diversity on evolving different types of walkers.

In all the above contributions the goal is to design efficient TL techniques that contribute to the effective transfer of knowledge between neural networks devised for different tasks. In this context, the last decade has witnessed the advent of new strategies to perform knowledge sharing between optimization tasks: the TO, introduced in Chapter 3 of this Thesis. This first technical contribution of the Thesis falls beneath the scope of EM, which gravitates on solving multiple problems simultaneously by exploiting their underlying synergistic relationships. The Thesis has already exposed that EM resorts to concepts and operators from Evolutionary Computation to implement multitasking optimization, with MFO as the most echoing formulation of the multitasking optimization problem considered by the research community working in EM. MFO was first introduced in [34] to formulate the transfer of knowledge between a set of simultaneously solved optimization problems. To realize this strategy, Multi Factorial Evolutionary Algorithm (MFEA) and its improved variant MFEA-II were proposed to adapt the parameters guiding the knowledge transfer among tasks, thereby reducing negative interactions between unrelated problems.

The work presented in this chapter adapts the search mechanisms of MFEA to multitask RL scenarios, including the adaptation of the parameters dictating the exchanged knowledge among tasks during the search. Specifically, the evolutionary approach designed and implemented for multitask RL, coined as A-MFEA-RL, builds on the logic behind MFEA, but includes new algorithmic ingredients that establish itself as a new algorithm. A-MFEA-RL is able to effectively operate over search spaces with large dimensions, such as those spanned by the trainable parameters (weights and biases) of neural networks. At the same time, operators of A-MFEA-RL are designed to foster layer-wise TL between several RL models evolved simultaneously. For this purpose, A-MFEA-RL is provided with the skills of every individual to decide which models and layers are more beneficial when shared for a given task.

In order to assess the performance of the proposed approach, an extensive experimentation is conducted seeking to answer three *research questions* around its adaptation capabilities, its comparison to other state-of-the-art approaches and its scalability with the number of tasks. Experiments are carried out over the *MT-10* and *MT-50* sets of robotic manipulation tasks comprising the Metaworld framework [517], both with and without initializing at random the initial state of each scenario at each episode. The comparison of A-MFEA-RL to state-of-the-art multitask RL is conclusive: A-MFEA-RL not only gets competitive results with respect to such *avant-garde* modeling counterparts, but also excels at effectively transferring knowledge among synergistically related RL tasks.

The rest of the chapter is organized as follows. In Section 4.1.1 the state of the art on evolutionary RL and multitask learning is analyzed, whereas Section 4.1.2 and Section 4.1.3 establishes fundamental definitions and concepts of RL and MFO. Section 4.1.2 describes in detail the proposed A-MFEA-RL approach, its search space and the rationale for the design of its operators. Next, Section 4.3 presents the experimental setup, poses the research questions to be answered, and discusses the obtained results. Finally, concluding remarks are given in Section 4.4.

4.1.1. Related Work

This section reviews the state of the art, departing from traditional approaches to multitask RL, and arriving at multitask evolutionary algorithms applied to multitask RL, stressing on how knowledge is transferred within each strategy.

There are multiple ways to share knowledge in non-evolutionary multitask RL. Multitask multi-headed networks [517, 513] are designed so that a part of the network is shared among all tasks. This shared part is in charge of providing a cross-task generalization capability to the overall model, whereas task-specific heads (typically, some few neural layers) are attached to the shared model. In this way, when facing a new task the model is trained by taking advantage of the knowledge persisted in the shared part of the model. Instead of sharing a part of a model, in [518] policies are encoded as modular neural networks, and annotated under *sketches* (a representation of a combination of sub-policies). When a new sketch is given, the previously stored knowledge is retrieved and used in the form of modules to make the agent resolve the task successfully. Other works such as DISTRAL [514] have studied the possibility to train multiple environments simultaneously by distilling the knowledge generated over each tasks to a shared policy. This shared policy is *fed back in* by virtue of a mechanism where the generated knowledge is transferred to the rest of RL models.

Shifting the focus towards evolutionary knowledge transfer, many approaches conduct genetic knowledge transfer based on different strategies.

In [348] and [519] memes are proposed as knowledge sharing enablers in a culture-inspired evolutionary optimization approach. Memes are considered building blocks that are able to communicate with each other via cultural operators, such as *assimilation* and *imitation*. In [519], the so-called Fusion Architecture for Learning and Cognition (FALCON) is introduced. In this work, memes represent neural networks, which are updated following the *meme internal evolution* and *meme external evolution* processes. The idea behind this approach is to make agents act as learners or teachers. When a learner chooses a teacher, *meme external evolution* is applied, and knowledge is transferred between them in the form of memes. Additionally, agents are allowed to learn individually using backpropagation by means of the *meme internal evolution*, by which memes are also updated. In [520] agents are sequentially evolved so that they can adapt to play a wide set of Atari games reusing knowledge acquired from previously evolved tasks. Also for Atari game playing, the work in [521] focuses on evolving task-specific agents and a multitask agent able to excel in a small set of Atari games simultaneously. A different approach to knowledge transfer is introduced in [522], in which a large-sized network is trained via backpropagation to learn multiple tasks. Tasks are embedded in the model, and critical forgetting is avoided by the evolution of agents that determine which subset of parameters are excluded from the gradient update. The approach is tested over a variety of supervised and RL tasks. A similar approach to the one proposed in this chapter is presented in [523], where knowledge is transferred among networks and the inputs and outputs are adapted while the internal structure is evolved via NE. The recurrent networks' structure is evolved by an adaptation of the EXAMM NE technique, and trained via backpropagation. Finally, evolved RNNs are used for transfer learning in two different scenarios, namely, coal-fired power plant (2 tasks) and aviation (3 tasks). Results showed that the evolved RNN architectures were beneficial for transfer learning among tasks.

It is often the case that EM methods deal with large-scale optimization problems due to the dimensions of the evolved networks. In these scenarios, a common approach is to split the problem to be solved into sub-problems of lower dimensionality. In [434], MFEA is used along with random embeddings to evolve large-scale continuous variable problems, performing knowledge transfer by using the implicit genetic transfer of MFEA operators. Therein, the problem is split into many sub-tasks, which are simultaneously evolved.

Next, works where MFEA is used for evolving problems related to Evolutionary Multitask RL are revised. In [17], the intrinsic knowledge transferred by MFEA is used to efficiently discover the best paths for multi-UAV path planning tasks. Similarly, in [407] multiple mobile agents for path planning tasks are simultaneously evolved by MFEA, which is nourished with external information (individual gradients) of the optimal direction for each agent. Finally, the works in [511, 314] introduce a modular network that is evolved by means of MFEA, with the avail of the implicit knowledge exchange over the unified search space. This search space is designed so that the evolved network can be encoded as a set of smaller networks (modules). The approach is tested over 4, 6 and 8-bit parity problems.

Despite the multiplicity of knowledge transfer methodologies reviewed above (multitask RL, memetic evolution, hybrid approaches, multifactorial optimization or NE), very few of them, if any, is able to automatically decide during the search the amount of knowledge to be transferred between tasks. In most approaches the relationship between tasks is assumed to be known beforehand, and/or the number of tasks evolved is small. The reason is the inherent difficulty of measuring those relationships without actually solving the tasks in question. Furthermore, it also turns impractical to manually discover these relations when the number of tasks grows. In [524], an attempt at gauging the shared inter-task knowledge in multitask RL is presented over the *Metaworld* and *CelebA* frameworks. Therefore, adapting the transfer of knowledge between models emerges as a natural step in the context of evolutionary multitask RL.

This chapter of the Thesis considers specifically the optimization of the trainable parameters (Problem 4) of deep neural networks used for RL. In this task, the outputs of the neural network encode the actions that conform the behavioral policy $\pi_{\boldsymbol{\theta}}$ of an agent when interacting with an environment `env`. Depending on the stimuli received by the network at its input (in a diversity of forms, from signals to images captured by the agent), its parameters $\boldsymbol{\theta}$ establish how the stimuli is translated to actions, mapping the behavior of the agents to successfully achieve a goal set for the environment at hand. Single-task settings have been usually approached by traditional RL algorithms such as proximal policy optimization or actor-critic methods. However, Evolutionary Computation can take a step further when solving for $\boldsymbol{\theta}$ thanks to their parallel nature, exploration skills and flexibility to be hybridized with problem-specific knowledge [283].

The optimization problem tackled in evolutionary Deep RL can be mathematically formulated as:

$$\max_{\boldsymbol{\theta}\in\theta} f_r(\mathtt{state}(\mathtt{env}, \mathtt{S}; \pi_{\boldsymbol{\theta}})), \tag{4.1}$$

where $f_r(\cdot)$ is defined, without loss of generality, as the median value of the reward function associated to the state reached when the agent applies the policy $\pi_{\boldsymbol{\theta}}$ on the environment `env` for a certain amount of steps `S`. In other words, the objective is to find the neural parameters $\boldsymbol{\theta}$ that make the policy $\pi_{\boldsymbol{\theta}}$ of the agent best solve the task portrayed in the environment `env`. It is important to note that the reward function is often determined by the environment and the task under consideration.

4.1.2. Multifactorial Evolutionary Algorithm

As stated in Section 3.3.2, MFEA is arguably the most renowned algorithm under the scope of multitask optimization. MFEA was proposed to simultaneously evolve a set of K optimization problems or tasks $\mathcal{T} = \{T_1, \ldots, T_K\}$, each defined by its particular boundaries, dimensions and fitness functions.

The main idea underlying this algorithm is to define an unified search space $\boldsymbol{\mathcal{X}}^U$, so that an encoded candidate $\mathbf{x} \in \boldsymbol{\mathcal{X}}^U$ can be decoded and evaluated into any of the K tasks, as it has been previously defined in Section 3.2.

Although the definitions introduced in Section 3.2 are common to all multifactorial optimization methods, the differential aspects of MFEA and its successor MFEA-II are their search operators (*Simulated Binary Crossover*, *Polynomial Mutation* and *Elitist Selection*), which rely on three main concepts:

- *Vertical cultural transmission*, under which each offspring individual inherits the skill factor τ^p of one of their parents.
- *Assortative mating*, by which candidates of same skill factor τ^p are more likely to exchange genetic material.
- *Transfer matrix* (RMP), composed by elements $rmp_{k,k'}$ (with $k, k' \in \{1, \ldots, K\}$) representing the probability to perform genetic knowledge transfer between two candidates with skill factors k and k' (defined in MFEA-II [374]).

Since its inception, MFEA and MFEA-II have gained increasing popularity within the community [525, 458, 392]. Nonetheless, they have been scarcely used for simultaneously training neural networks [511]. This is mainly due to two causes, both related to the design of the unified space $\boldsymbol{\mathcal{X}}^U$. First, the RMP matrix featured by MFEA-II is not well suited to allow for knowledge transfer over unified search spaces with a layered structure underneath, but rather defines the transfer procedure uniformly over the entire genotype of solutions. On the other hand, architectural differences between the networks to be evolved for every task, such as layers' number and size, must be taken in account when designing the unified space. Both considerations are addressed in the proposed A-MFEA-RL.

4.1.3. Transfer Learning

As stated in Section 1.1, TL refers to the way in which knowledge learned by a model when addressing a certain task is reused as the starting point for the construction of a model for another task. There are many approaches that conduct TL between neural networks under diverse strategies, namely, *instance-based*, *mapping-based*, *adversarial-based* and *network-based* [526]. Network-based approaches have been widely applied in image recognition, single-task RL [527, 528] and multitask RL [512, 514]. These approaches hinge on reutilizing pretrained parts of a neural network (layers) used to solve a task for expediting the convergence of another model for a second task. To this end, some layers from the network of the source task are copied to the model to be learned for the target domain, for which the latter must ensure strict neural architectural similarities with the network from which the knowledge is imported. Once copied, parameters of such transferred layers are often kept fixed, whereas the remaining layers are trained regularly via gradient backpropagation. The contribution of the transferred knowledge is constrained by the amount of information that tasks from the source and target domains have in common. The more similar the domains are, the more knowledge can be transferred between them. When this knowledge exchange entails an improvement of the target task, the transfer is said to be *positive*. Contrarily, knowledge transfer from an unrelated source domain can hinder the training process in the target domain, resulting in what is known as *negative transfer* [529, 530].

Even if TL is conceptually simple to implement, there is no consensus on how to measure the similarity between two tasks. This makes the decision on how much information to transfer uninformed, and often achieved as a result of successive performance-driven trials. Despite this caveat, TL can be very useful for RL [531], allowing agents to adapt quickly to new environments. When dealing with extremely complex behavioral policies, agents often require gradual RL, by which the agent is sequentially trained to learn progressively more complex tasks. Past knowledge (policies) is reused by means of TL towards achieving good policies for the most convoluted task faster [532]. The A-MFEA-RL approach described in what follows aims indeed at this goal, by providing core modifications of the unified space, crossover operator and knowledge transfer criteria towards realizing an adaptive network-based TL mechanism.

4.2. Proposed approach: A-MFEA-RL

This section delves into A-MFEA-RL, the proposed evolutionary multitasking approach capable of evolving multiple RL models simultaneously, disregarding whether the tasks for which they are devised are related to each other. The section stresses on the intuition followed to endow A-MFEA-RL with the capability to avoid negative transfers and favor the exchange of knowledge between synergistic tasks. This specific focus on the adaptability of A-MFEA-RL is complemented with specific details on the design of the unified space (Section 4.2.1), the adaptation of the crossover operator (Section 4.2.2) and how adaptive TL is updated during the multitasking search process (Section 4.2.3). A reference pseudocode of A-MFEA-RL is given in algorithm 1.

4.2.1. Design of the Unified Search Space

As has been mentioned in Section 4.1.2, knowledge transfer between simultaneously evolved RL models is carried out over the unified space $\boldsymbol{\mathcal{X}}^U$. This unified space must be designed properly to account for the distribution and size of the neural networks it represents when decoded for every RL task. Therefore, the goal is to derive a unified space in which model-based transfer learning can be easily implemented.

A-MFEA-RL is designed so that K different models, that behave as mappers between policies and environments, are simultaneously evolved. Let

$\mathcal{M} = \{M^{T_1}_{\boldsymbol{\theta}^p}, M^{T_2}_{\boldsymbol{\theta}^p}, \ldots, M^{T_K}_{\boldsymbol{\theta}^p}\}$ be the set of models decoded from the evolved parameters $\boldsymbol{\theta}^p \equiv \mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$ for solving RL tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$, respectively. Any of these models $M^{T_k}_{\boldsymbol{\theta}^p}$ can be defined as a set of layers $\{L^1_k, L^2_k, \ldots, L^{\varphi_k}_k\}$, where φ_k stands for the number of layers of the model selected to undertake task T_k. The unified space must be designed so that it represents the layers of all models $M^{T_k}_{\boldsymbol{\theta}^p}$ $\forall k \in \{1, \ldots, K\}$, allowing for the transfer of knowledge among tasks. Therefore, an encoded candidate $\mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$ is divided in two parts:

- A set of φ_{sh} layers that are common to all models, defined as

$$\boldsymbol{\mathcal{L}}^{sh} = \{L^{sh,1}, \ldots, L^{sh,\varphi_{sh}}\}, \tag{4.2}$$

  where φ_{sh} is the number of shared layers, and $|L^{sh,\ell}| = \max_k |L^\ell_k|$ for $\ell \in \{1, \ldots, \varphi_{sh}\}$, with $|\cdot|$ denoting number of parameters. In what follows superscript ℓ will be used to denote layer index.

- A second set of layers $\boldsymbol{\mathcal{L}}^{sp} = \{\boldsymbol{\mathcal{L}}^{sp}_k\}_{k=1}^K$, which is composed by the concatenation of the layers $\boldsymbol{\mathcal{L}}^{sp}_k = \{L^{sp,1}_k, \ldots, L^{sp,\varphi_{sp,k}}_k\}_{k=1}^K$ that are specific for every task T_k, and hence do not take part in the transfer learning mechanism. Here, $\varphi_{sp,k}$ denotes the number of task-specific layers for task T_k, such that $\varphi_{sh} + \varphi_{sp,k} = \varphi_k$.This permits to discern between transferable layers $(1 \leq \ell \leq \varphi_{sh}\})$ and non-transferable, task-specific layers (corr. $\ell > \phi_{sh}$).

Therefore, the dimensionality of the parameters evolved by A-MFEA-RL is given by:

$$|\mathbf{x}^p| = \sum_{\ell=1}^{\varphi_{sh}} |L^{sh,\ell}| + \sum_{k=1}^{K} \sum_{\ell'=1}^{\varphi_{sp,k}} |L^{sp,\ell'}_k|, \tag{4.3}$$

where the first term corresponds to the aggregate number of parameters of the shared layers, and the second term represents the sum of the parameters of all individual layers that do not participate in the knowledge sharing part.

Now that the unified search space is defined, it is time to define how parameters belonging to the layers $\{L^1_k, \ldots, L^{\varphi_k}_k\}$ of the model devised for task T_k are decoded from $\mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$. Since candidates may represent neural networks of different number of layers and/or neurons within each layer, the location of task-specific parameters inside the encoded individual $\mathbf{x}^p$ has to be carefully designed so that weights from a model and biases from another do not overwrite each other when TL is performed. Otherwise, the transfer will be incoherent and counterproductive for the search [533].

To circumvent this issue, a clear distinction between weights $\mathbf{w}^\ell_k$ and biases $\mathbf{b}^\ell_k$ of layer L^ℓ_k is defined, so that the transfer of knowledge considers the potentially different network architectures among tasks. To mathematically define this tailored decoding process, two mapping functions are designed, these functions translate $\mathbf{x}^p$ to $\mathbf{w}^\ell_k$ and $\mathbf{b}^\ell_k$ for $\ell \in \{1, \ldots, \varphi_{sh} + \sum_{k=1}^K \varphi_{sp,k}\}$, expressed as $\lambda^\ell_{k,\mathbf{w}}(\cdot)$ and $\lambda^\ell_{k,\mathbf{b}}(\cdot)$, such that $\lambda^\ell_{k,\mathbf{w}}(\mathbf{x}^p) = \mathbf{w}^\ell_k$ and $\lambda^\ell_{k,\mathbf{b}}(\mathbf{x}^p) = \mathbf{b}^\ell_k$.

Algorithm 1: Proposed A-MFEA-RL.

```
1  $\mathcal{T} \leftarrow$ set of tasks $\{T_1, T_2, \ldots, T_K\}$ and $K = |\mathcal{T}|$
2  Define number of individuals per task as $P_k$
3  Define unified search space $\boldsymbol{\mathcal{X}}^U$ and mapping functions $\lambda^{\ell}_{k,\mathbf{w}}(\cdot)$ and
     $\lambda^{\ell}_{k,\mathbf{b}}(\cdot)$ for every task $T_k$ *(Section 4.2.1)*
4  Generate a population $\mathcal{P}$ of $P = \sum_{k=1}^{K} P_k$ candidates $\{\mathbf{x}^p\}_{p=1}^{P}$, with
     $\mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$
5  Assign every $\mathbf{x}^p$ a skill factor $\tau^p \in \{1, \ldots, K\}$, maintaining the
     representation of all tasks in the population as per $\{P_k/P\}_{k=1}^{K}$
6  Set $rmp^{\ell}_{k,k'} = rmp_{ini}$ and $rmp^{\ell}_{k,k} = 1$ $\forall \ell$ and $k \neq k'$
7  Evaluate every $\mathbf{x}^p$ over task $T_{\tau^p}$ *(Section 4.2.4)*
8  Set gen = 1
9  while $gen \leq maxGen$ do
10     Couple all individuals $\{\mathbf{x}^p\}_{p=1}^{P}$ in pairs at random
11     Offspring population $\mathcal{Q} \leftarrow \emptyset$
12     Set $C^{\ell,+}_{k,k'} = C^{\ell,-}_{k,k'} = 0$
13     for each coupled pair of individuals $\mathbf{x}^p$, $\mathbf{x}^{p'}$ do
14         Select a subset of the shared layers $\mathcal{L}^{sh}$ for crossover as per
             $rmp^{\ell}_{\tau^p,\tau^{p'}}$
15         if at least one layer to be exchanged then
16             $\mathbf{x}^p_o, \mathbf{x}^{p'}_o \leftarrow$ SBX$(\mathbf{x}^p, \mathbf{x}^{p'})$ *(Section 4.2.2)*
17             $\tau^p_o \leftarrow$ randomChoice$(\tau^p, \tau^{p'})$
18             $\tau^{p'}_o \leftarrow$ randomChoice$(\tau^p, \tau^{p'})$
19         else
20             $\mathbf{x}^p_o \leftarrow$ polMut$(\mathbf{x}^p)$, $\mathbf{x}^{p'}_o \leftarrow$ polMut$(\mathbf{x}^{p'})$
21             $\tau^p_o \leftarrow \tau^p$, $\tau^{p'}_o \leftarrow \tau^{p'}$
22         end
23         Evaluate $\mathbf{x}^p_o$ on $T_{\tau^p_o}$ *(Section 4.2.4)*
24         Evaluate $\mathbf{x}^{p'}_o$ on $T_{\tau^{p'}_o}$
25         Update $C^{\ell,+}_{k,k'}$, $C^{\ell,-}_{k,k'}$ for $k = \tau^p$ and $k' = \tau^{p'}$
26         Store offspring: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathbf{x}^p_o, \mathbf{x}^{p'}_o\}$
27     end
28     Create combined population $\mathcal{P}' \leftarrow \mathcal{P} \cup \mathcal{Q}$
29     Build the new population $\mathcal{P} \subset \mathcal{P}'$ by retaining the best individuals
         as per their scalar fitness $\varphi^p$, and maintaining the task
         distribution $(\{P_k/P\}_{k=1}^{K})$
30     Update $rmp^{\ell}_{k,k'}$ based on $C^{\ell,+}_{k,k'}$ and $C^{\ell,-}_{k,k'}$ $\forall k, k', \ell$ *(Section 4.2.3)*,
         and set gen = gen + 1
31 end
32 Return the best individual in $\mathcal{P}$ for each task $T_k$
```

Clearly, the total number of parameters of the ℓ-th layer of the model constructed for task T_k is given by $|L^{\ell}_k| = |\mathbf{w}^{\ell}_k| + |\mathbf{b}^{\ell}_k|$. It should be also noted

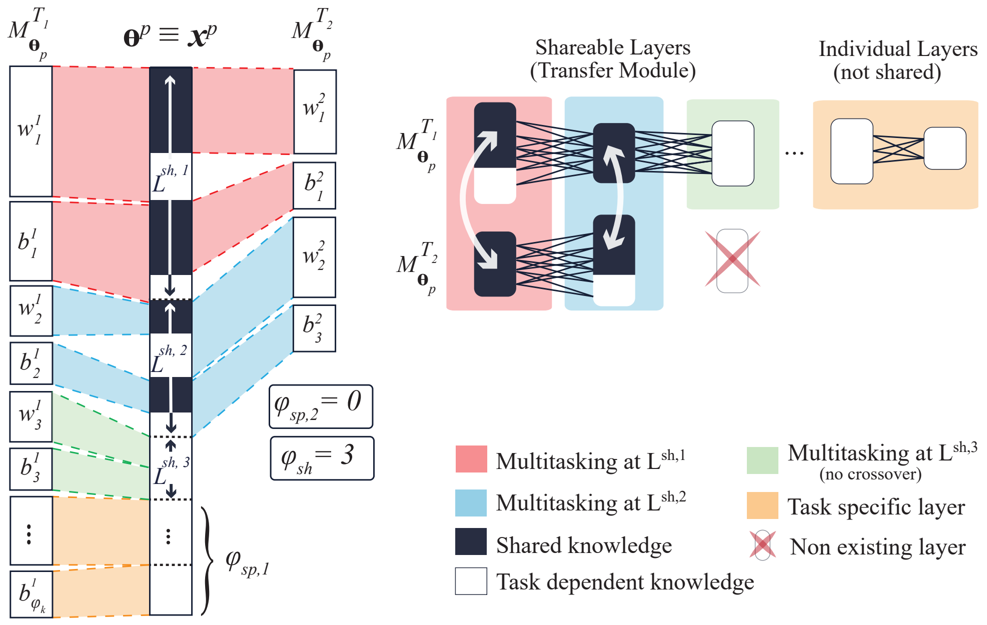

Figure 4.1: Schematic diagram illustrating: (left) how the unified search space $\mathcal{X}^U$ is structured in a layer-wise manner, including how it is decoded for two different tasks T_1 and T_2; and (right) how those decoded individuals yield two RL models with partly shared weights and biases.

that for some layers in $\mathbf{x}^p$, $|\mathbf{w}^\ell_k| = |\mathbf{b}^\ell_k| = 0$, i.e., not all mapping functions impose that the model for the task at hand should be assigned weights and biases from all layers represented in $\mathbf{x}^p$. This holds for T_k and any task-specific layer $L^{sp,\ell}_{k'}$ with $k' \neq k$. Summarizing, the mapping functions $\lambda^\ell_{k,\mathbf{w}}(\cdot)$ and $\lambda^\ell_{k,\mathbf{b}}(\cdot)$ allow translating every candidate $\mathbf{x}^p$ to the specific network architecture for every task.

Following the schematic example shown in Figure 4.1 for $K = 2$ tasks, first can be noted that weights and biases in every layer represented in $\mathbf{x}^p$ are isolated from each other by the aforementioned mapping functions. Furthermore, it is straightforward to see that when $\ell \in \{1, \ldots, \varphi_{sh}\}$ (namely, the *shared* layers), weights $\mathbf{w}^\ell_k$ and biases $\mathbf{b}^\ell_k$ decoded for task T_k overlap with those defined for task $T_{k'}$. Hence, this decoding process eases the transfer of knowledge among tasks via TL, as task-specific models, when decoded and built from $\mathbf{x}^p$, share part of their weights and biases in the shared layers. Furthermore, another advantage of the designed unified search strategy is that, once layers, weights and biases are arranged in the genotype of $\mathbf{x}^p$, the evolved networks are not restricted to be architecturally identical for them to exchange information and achieve effective TL.

4.2.2. Search Operators

As in MFEA and MFEA-II, A-MFEA-RL resorts to the *Simulated Binary Crossover* (SBX) operator, which has been adapted to work in a layer-wise fashion and designed to transfer only valid information. Following the notation introduced in the previous section, let $\mathbf{x}^p$ and $\mathbf{x}^{p'}$ be two candidates encoded in the unified search space $\boldsymbol{\mathcal{X}}^U$, with τ^p and $\tau^{p'}$ indicating their skill factors. Assuming that $\tau^p \neq \tau^{p'}$, the potentially different network architectures makes it necessary to discriminate the amount of knowledge $\mathbf{x}^p$ and $\mathbf{x}^{p'}$ can effectively share at each layer.

To this end, the SBX operator is only applied to the decision variables belonging to the set of shared layers $\boldsymbol{\mathcal{L}}^{sh}$ of both $\mathbf{x}^p$ and $\mathbf{x}^{p'}$. However, not all shared layers inside $\boldsymbol{\mathcal{L}}^{sh}$ are selected for crossover. Instead, those shared layers whose exchange has been shown to be beneficial during the search should be selected with high probability for crossover. Conversely, the algorithm should avoid selecting those shared layers that, when involved in previous mating processes, have yielded worse offspring than their parents. Furthermore, the different skill factors τ^p and $\tau^{p'}$ of the individuals should also influence which shared layers should be eligible for the crossover operator.

In order to implement this two-fold adaptability, A-MFEA-RL resorts to a modified version of the so-called transfer matrix RMP introduced in MFEA-II [374]. In its seminal form, each entry $rmp_{k,k'}$ of this $K \times K$ matrix establishes the probability of two individuals with skill factors $\tau^p = k$ and $\tau^{p'} = k'$ exchange knowledge through crossover. However, in A-MFEA-RL the RMP matrix is scaled up to a tensor of dimensions $K \times K \times \boldsymbol{\mathcal{L}}^{sh}$ so as to model not only the relationships between tasks, but also between shared layers. Accordingly, entries in the RMP matrix are now denoted as $rmp^{\ell}_{k,k'}$, where $\ell \in \{1, \ldots, \varphi_{sh}\}$ and $k, k' \in \{1, \ldots, K\}$. These entries of RMP are used in A-MFEA-RL as the probability that the optimization variables belonging to shared layer ℓ are selected for crossover between a pair of parent individuals $\mathbf{x}^p$ and $\mathbf{x}^{p'}$ with skill factors τ^p and $\tau^{p'}$, respectively.

Based on concepts from TL, the amount of information two individuals qualified for the same skill task T_k should exchange as much information as possible. This is imposed by forcing $rmp^{\ell}_{k,k} = 1.0 \ \forall k, \ell$. Another restriction arises from the case where the neural network decoded for task T_k comprises less layers than the amount of shared layers (e.g. when $\varphi_k < \varphi_{sh}$. In this case, since shared layers beyond φ_k do not contain any knowledge about the task T_k, they should not be shared with other individuals. Thereby, $rmp^{\ell}_{k,k'}$ and $rmp^{\ell}_{k',k}$ are fixed to 0 for any k' and $\ell \in \{\varphi_k + 1, \ldots, \varphi_{sh}\}$.

On the other hand, the mutation operator adopted in A-MFEA-RL is the classical *polynomial mutation*, which provides an additional level of diversity during the search and helps the algorithm escape from local optima. It is important to note that unlike the original MFEA, crossover and mutation operators are not sequentially (i.e. crossover + mutation) applied during offspring generation. Instead, candidates are only recombined or mutated depending on whether any layer has been selected for crossover among the parent individuals. The reason for this modified application of evolutionary operators is that the mutation of previously mated candidates may induce noise into the knowledge transferred among individuals, leading to a worse convergence of the overall algorithm.

With the modified crossover operator described in this section, A-MFEA-RL is able to effectively implement TL between the networks decoded from the unified search space $\boldsymbol{\mathcal{X}}^U$ for every task. One of the major advantages resulting from this specialized crossover is that the knowledge transfer is held coherently even if such networks comprise different number of layers and/or varying number of neurons. However, an additional degree of adaptation is still needed for the crossover operator to promote the exchange of knowledge between tasks found out to complement each other during the search. This is realized by means of a procedure devised to update the adaptive knowledge transfer mechanism performed by the modified crossover operator, which is detailed next.

4.2.3. Adaptive Knowledge Transfer

As stated previously, in A-MFEA-RL the modified SBX crossover operates on the basis of a RMP matrix, whose entries $rmp^{\ell}_{k,k'}$ establish the probability of two individuals with different skill factors to select a shared layer for their mating. This allows mimicking the traditional way to of implementing TL between neural networks, which relies on the copy of part of the parameters of a source network to a target network, the latter getting advantage of this transferred knowledge. However, this probabilistic knowledge transfer criterion should adapt the values of $rmp^{\ell}_{k,k'}$ during the search, so that the updated values would favor future crossovers when proven to contribute to a successful knowledge transfer (i.e. better offspring that their parents).

Proceeding now with the explanation of how $rmp^{\ell}_{k,k'}$ values are updated, first, whenever two candidates $\mathbf{x}^p$ and $\mathbf{x}^{p'}$ with skill factors $\tau^p = k$ and $\tau^{p'} = k'$ are selected for crossover, a φ_{sh}-length binary mask $\mathbf{m} = \{m^{\ell}\}_{\ell=1}^{\varphi_{sh}}$ is computed to indicate the shared layers that are chosen for the knowledge transfer process. This mask is computed as:

$$m^{\ell} = \begin{cases} 1 & \text{if } rand < rmp^{\ell}_{k,k'}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.4}$$

where $rand$ is the realization of a continuous random variable uniformly distributed over $\mathbb{R}[0,1]$. Then, only weights and biases of those shared layers for which $m^{\ell} = 1$ are selected for standard SBX crossover. This process is done independently for every pair of individuals selected for mating. After mating, two counters $C^{\ell,+}_{k,k'}$ and $C^{\ell,-}_{k,k'}$, both initialized to 0 at the beginning of each generation, count the times the fitness of an offspring individual is better $(+)$ or worse $(-)$ than that of its parents. Once all coupled pair of parent individuals have been processed, the values of the RMP are updated as:

$$rmp^{l}_{k,k} \leftarrow \texttt{clip}^{rmp_{max}}_{rmp_{min}}\Big(rmp^{l}_{k,k} - \alpha \cdot C^{\ell,-}_{k,k'} + \beta \cdot C^{\ell,+}_{k,k'}\Big) \tag{4.5}$$

where $\alpha, \beta \in \mathbb{R}(0,1)$ represent the increase (decrease) that a positive (negative) knowledge transfer imprints on the value of $rmp^{\ell}_{k,k'}$ value. In order to avoid that $rmp^{\ell}_{k,k'}$ eventually reaches 0 $\forall k', \ell$ for a given task T_k, lower (rmp_{min}) and upper (rmp_{max}) bounds are imposed in the value of $rmp^{\ell}_{k,k'}$ by means of a clipping function $\texttt{clip}^{b}_{a}(x) \doteq \max\{b, \min\{a, x\}\}$. This prevents the algorithm from falling into a state where a task does not share/import any knowledge.

4.2.4. Evaluation Criterion

Evolved candidates are evaluated in A-MFEA-RL by assessing how the decoded neural networks, which represent the behavioral model of the agent, performs for the task T_k defined in its environment. A reward function $f_r(\cdot)$ provides the algorithm with a numerical value that quantifies the *quality* of the agent when undertaking the task. This reward function takes a central role in the definition of the loss function defined for training neural RL models via gradient backpropagation. The achievement of high reward values is directly linked to the successful completion of the task. Therefore, the choice of the reward function value reached by the decoded model for a maximum number of steps $\texttt{S}$ seems a reasonable choice.

Nevertheless, relying on a single reward score achieved by the agent for the task T_k could not be representative enough of its performance. Therefore, each candidate is evaluated over $\texttt{E}$ different episodes, so that the fitness value $F_k(\mathbf{x}^p_k)$ of the individual $\mathbf{x}^p_k$ decoded from $\mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$ for task T_k is computed as the median value of the $\texttt{E}$ reward values obtained after $\texttt{S}$ steps simulated for each episode.

4.3. Experimental Setup & Results

As a result of its tailored design, A-MFEA-RL addresses the goal of simultaneously evolving multiple RL agents, automatically discovering and exploiting inter-task synergies, and effectively avoiding negative knowledge transfer. In order to assess and validate these claims, this section aims to present and discuss on the results of an extensive simulation setup designed to give an informed response to five *research questions* (RQ):

- RQ1: *Can A-MFEA-RL efficiently exploit positive inter-task synergies, and elude negative knowledge transfer?*
- RQ2: *Does A-MFEA-RL perform competitively compared to existing multitask RL approaches?*
- RQ3: *Does A-MFEA-RL scale up nicely when more RL tasks are simultaneously evolved?*
- RQ4: *Does the adaptive knowledge transfer of A-MFEA-RL provide any gain with respect to other baseline knowledge transfer methods?*
- RQ5: *Is the computational cost of A-MFEA-RL competitive with respect to other multitask RL methods?*

While answers given to RQ1 and RQ3 may serve as a way to validate the performance of the algorithm and the concepts introduced throughout the paper, RQ2, RQ4, and RQ5 are devoted to the comparison of A-MFEA-RL to other state-of-the-art multitask RL methods:

- *Multitask Proximal Policy Optimization* (M-PPO) [534].
- *Multitask Trust Region Policy Optimization* (M-TRPO) [535].
- *Task Embeddings* (TE) [536].
- *Multitask Soft Actor Critic* (M-SAC) [513].
- *Multitask Multi-head Soft Actor Critic* (M-M-SAC) [513].
- *Shallow Multitask Reinforcement Learning with Soft Modularization* (S-MRL-SM) [537].
- *Deep Multitask Reinforcement Learning with Soft Modularization* (D-MRL-SM) [537].

The experimentation is built upon the *Metaworld* framework [517], which implements a set of 50 robot arm based tasks (Figure 4.2) modeled and run over the MuJoCo physics simulator [538]. Tasks T_k in *Metaworld* are designed so that all of them feature some degree of similarity to each other, while being sufficiently diverse to ensure meaningful studies related to multitask learning. Further details about the environment and tasks can be found in [517]. These *Metaworld* environments allow evaluating the performance of developed RL agents in a given multitasking configuration, yielding a comprehensive playground where to gauge the performance of the proposed A-MFEA-RL. This evaluation is done in terms of the average success rate achieved by each approach over $\mathtt{E} = 300$ test episodes.

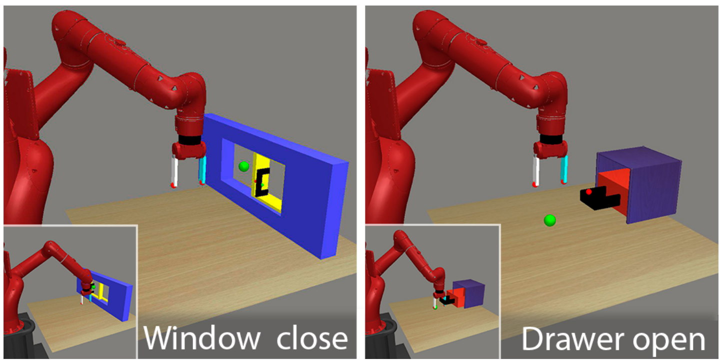

Figure 4.2: Images illustrating the `window close` and `drawer open` tasks of the *Metaworld* framework. The ending state of a successfully completed episode is shown in the nested image on the bottom left corner of every plot.

Among the totality of 50 environments embedded in *Metaworld*, some of them are relatively more complicated than others for a RL agent to solve them successfully. Accordingly, the 50 environments are divided in `EASY`, `MEDIUM` and `HARD`, based on the number of subtasks needed for the completion of their corresponding task. The definition of this level of complexity permits to define three different architectures of the neural network to be evolved by A-MFEA-RL, which dictates how individuals $\mathbf{x}^p$ from the unified search space $\boldsymbol{\mathcal{X}}^U$ are decoded. Details on these task complexity-dependent architectures are listed in Table 4.1, along with their average number of trainable parameters (weights and biases).

Table 4.1: Network architectures for every task complexity level. All layers are fully connected comprising the indicated number of neurons, whereas + denotes layer concatenation.

Complexity	**Neural architecture**	**Parameters**
Easy	256 + 128 + 128 + output layer	$\sim$ 54K
Medium	256 + 128 + 128 + 128 + output layer	$\sim$ 70K
Hard	256 + 128 + 128 + 128 + 128 + output layer	$\sim$ 87K

A summary of the parameters of A-MFEA-RL configured for all simulations is listed in Table 4.2. The value of *start_rmp* is high to ensure a high number of transfers in early stages of the evolutionary process and thereby, feed the $rmp^{\ell}_{k,k'}$ tensor with potentially successful crossovers. The imbalanced values imposed on alpha and beta serve as a heuristic way to grant more importance to synergistic relationships than to negative knowledge transfers. The selected population size P_k, number of generations maxGen and the number of evaluation episodes `E` leave the overall number of processed samples $(6 \cdot 10^6)$ significantly below those used in the rest of multitask RL counterparts $(15 \cdot 10^6)$. Parameter values for the rest of multitask RL methods considered in RQ2, RQ3 and RQ5 are equal to those utilized in the publications where they were first presented.

Table 4.2: Parameters used in the experiments.

Parameter	**Value**	**Parameter**	**Value**
P_k	60 for all tasks (i.e. $P = 60K$)	rmp_{min}	0.15
`E`	300 episodes	rmp_{max}	0.95
maxGen	600 (RQ1), 1000 (RQ2 to RQ5)	α	0.01
start_rmp	0.80	β	0.10

The experimentation runs over a computer equipped with four Intel Xeon Gold 5120 processors running at 2.2 GHz and 1 Tb of RAM. For the sake of computational efficiency and to fully leverage the availability of multiple processing cores, a highly parallel implementation of A-MFEA-RL has been used, which has been made available in a public repository (`https://git.code.tecnalia.com/aritz.martinez/a-mfea-rl`) along with the scripts generating the results discussed in the paper.

Considering this simulation setup, the next sections elaborate on empirical evidence aimed to address the five research questions formulated above.

4.3.1. RQ1: Can A-MFEA-RL Efficiently Exploit Positive Inter-task Synergies and Elude Negative Knowledge Transfer?

The main goal of RQ1 is to verify whether A-MFEA-RL is able to effectively promote the exchange of information (through crossover) between synergistic tasks, as well as to avoid negative knowledge transfer among tasks that are unrelated. To this end, the scenario designed is simple and it is composed by 5 different tasks, each repeated twice, so that a total of $K = 10$ tasks are simultaneously evolved. The reason of this simplistic design is two-fold. On the one hand, it allows for clear insights shed over the purpose of the experiment (transferability). On the other hand, the fact that tasks are duplicated ensures that there are sets of tasks for which the algorithm should promote maximum knowledge transfer. This should be clearly noticed when analyzing the output of A-MFEA-RL. We hereafter denote such duplicated tasks as *twin tasks*.

In Figure 4.3, inter-task and intra-task relationships are represented quantitatively as a bubble plot matrix linking every pair of tasks. Specifically, the radius of every bubble plot is proportional to the number of successful crossovers wherein an individual specialized for the *helping task* (indicated in the X axis) improved the fitness of another individual skilled for the *helped task* (corr. Y axis). Each of such bubble plots is further divided depending on the relative number of times every neural layer participated in the knowledge transfer enabled by the crossover. Bubble plots in the diagonal indicate the relative number of times each individual improved via inter-task or intra-task crossover (namely, between individuals specialized in the same task).

First of all, Figure 4.3 evinces that A-MFEA-RL promotes mating individuals belonging to every pair of twin tasks, fact that validates the capability to find synergistic tasks sought for the algorithm. Indeed, the crossover between twin tasks is the most effective among the totality of considered tasks, as shown by the highest radii of bubbles relating a task with its twin. Interestingly, the bubble plot matrix unveils some other synergistic relationships between tasks that conform to intuition and as such, deserve to be mentioned. To begin with, the `drawer-close` environment benefits from mating with `door-open` and `window-close`, as the latter two comprise approaching and pushing movements similar to those needed for closing the drawer. Likewise, pausing at `window-close`, the environment is prone to leverage the crossover with individuals skilled at `drawer-close` and `door-open`. These synergies are also observed between `drawer-close` and `door-open`, further validating the proficiency of A-MFEA-RL in encouraging transfer learning between positively related environments.

When focusing on how the above knowledge exchange is distributed over layers of the networks involved in the transfer, just recall that A-MFEA-RL extended the definition of the RMP matrix towards deciding which layers to enter the crossover operation. This is noted, for instance, in the transfer between `button-press` twin tasks, where the first layer participates in most

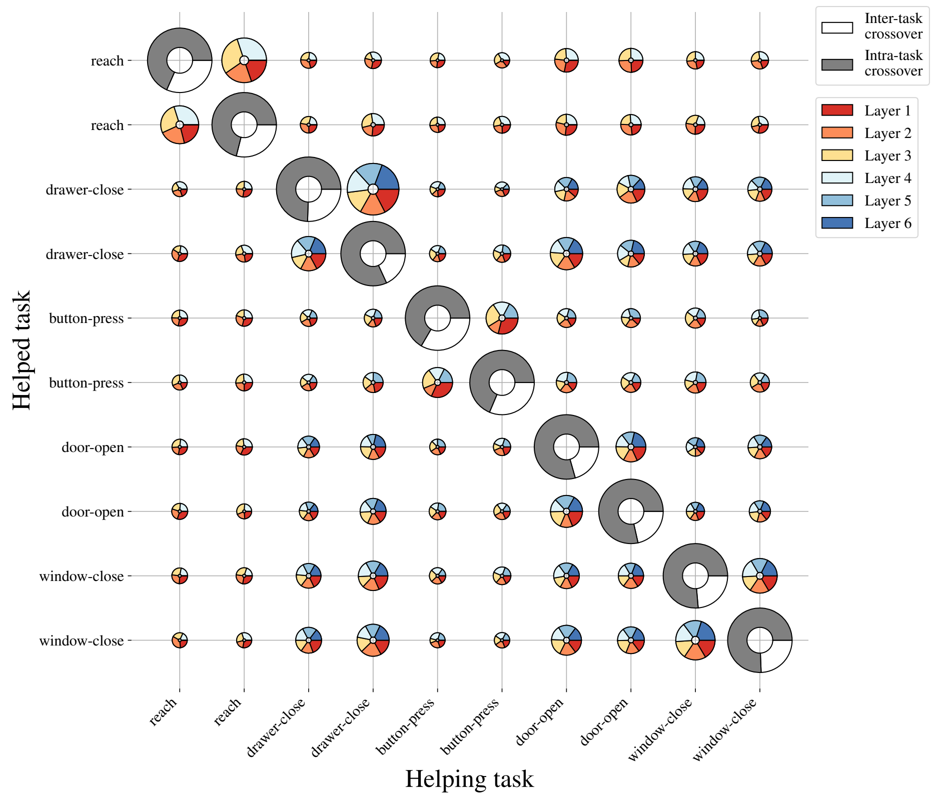

Figure 4.3: Bubble plot matrix showing the amount of knowledge transferred between tasks involved in RQ1, for every layer of the neural networks of the evolved agents.

of the crossovers. On the contrary, for the `reach` twin tasks A-MFEA-RL tends to perform crossover over the third and fourth layers. Given the relative simplicity of this task, this suggests that most of the other tasks contribute coherently to the knowledge required for the first layers of the network (specially `door-open`). Therefore, crossovers among individuals of twin `reach` tasks aim to complement the knowledge not transferred from the rest of tasks.

Nevertheless, this transferability analysis should also account for the success of the evolved agents when facing the RL tasks considered in RQ1. To this end, Table 4.3 summarizes the success rates (in %) of all environments, where a noteworthy effect can be observed regarding `reach` (the less successfully solved environment). The goal in this environment is to reach a goal whose position changes between episodes. As many other tasks require approaching an object before performing an action and completing the episode, intuitively `reach` should synergistically contribute to the progressive improvement of other environments. Surprisingly, the only task aided by `reach` is its own twin task. Bearing in mind its worse success rate, this unexpected result posits that the information stored in the population is more valuable when its contained individuals achieve high levels of success. Thus, individuals specialized for tasks that converge slower, when mated, contribute to the produced offspring in terms of exploration. As they progressively achieve higher rewards and success rates, such individuals start contributing with valuable information to the mating process.

Table 4.3: Success rate of tasks evolved in RQ1.

Task name	Twin #1	Twin #2
`reach`	86.3%	86.6%
`drawer-close`	100%	100%
`button-press`	97.6%	99.3%
`door-open`	100%	100%
`window-close`	100%	100%

To sum up, they can be witnessed several clear synergies between tasks in this first experimental setup, not only between twin tasks, but also a/cross different tasks that can be intuitively expected to maintain a degree of relationship (e.g. `window-close` and `drawer-close`). Quality and transferability have also been proven to be coupled to each other, in that knowledge transfer allows for improved specialization of the destination (*helped*) task whenever the model of the source (*helping*) task solves its environment satisfactorily.

4.3.2. RQ2: Does A-MFEA-RL Perform Competitively Compared to Existing Multitask RL Approaches?

We turn the focus on RQ2, which aims to compare A-MFEA-RL to other state-of-the-art RL approaches suited for multitask environments. For this purpose, the multitask scenario designed is similar to the one used in [537], extending the so-called *MT-10* problem proposed in [517] so that the goal and object positions are set at random between different episodes. This extension leads to two scenarios: 1) the fixed *MT-10*, comprising 10 different environments whose tasks are fixed; and 2) the randomized *MT-10* (labeled as *MT-10-R*), which permits to evaluate the capability of the evolved agents to generalize to potentially unseen environments. The elements that change between episodes in the randomized *MT-10-R* vary depending on the task, and are described in [517].

This discussion follows the same flow as the one made in the previous subsection for RQ1. Accordingly, the *helping* tasks that contributed most for the improvement of individuals specialized in other *helped* tasks are analyzed. From Figure 4.4 it can be observed that `drawer-close`, `button-press`, `window-close` and `door-open` are the top tasks in terms of successful crossovers with the rest of tasks. This fact is in full accordance with the conclusions drawn in [524], where some *Metaworld* tasks (`reach`, `push`, `button-press-topdown`, `window-open` and `window-close`) are co-trained in different combinations to quantitatively analyze the degree of overlap of their knowledge. In this recent study, such tasks are found to be very positively related to each other, which is also concluded from Figure 4.4.

Moreover, `reach` and `push` clearly prefer to mate with `button-press` or `window-close` rather than exchanging information with `window-open`, facts that are also among the findings reported in [524].

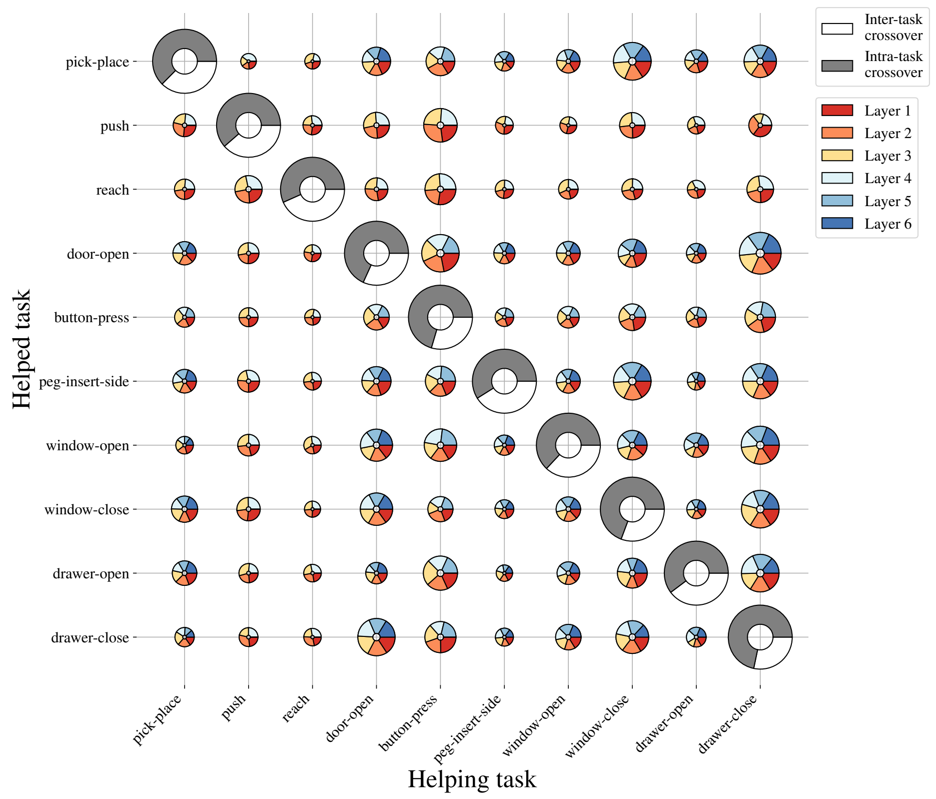

Figure 4.4: Bubble plot matrix showing the amount of knowledge transferred between tasks in *MT-10-R*.

Interestingly, there are two environments for which A-MFEA-RL was unable to find good policies, yielding null success rates: `pick-place` and `peg-insert-side`. By analyzing closely the synergies of the rest of tasks with these two complex environments, it is found that almost any other task contributes with the exchange of its knowledge. Among the tasks helping these two environments, `window-close` and `drawer-close` are the ones generating more positive transfers. In addition, conclusions held in RQ1 also apply to this second setup, as per the synergies noted between `drawer-close`, `window-close` and `door-close`.

Following the previous discussion, Table 4.4 lists the average success rates achieved by A-MFEA-RL in the *MT-10* and *MT-10-R* scenarios. Also are included for comparison the average success rates achieved by M-PPO, M-TRPO, TE, M-SAC, M-M-SAC, S-MRL-SM and D-MRL-SM over the environments. To this comparison, they are also added the results of a single-task Soft Actor Critic (SAC) agent trained over $6 \cdot 10^6$ samples per task and the neural architectures in Table 4.1 for guaranteeing a fair comparison to *A-MFEA-RL*. These results are complemented by those in Table 4.5, which shows the success rates for every task and the top three algorithms in this benchmark (i.e., S-MRL-SM, D-MRL-SM and A-MFEA-RL).

Table 4.4: Success rate of tasks evolved for RQ2: MT-10 and MT-10-R.

Algorithm	**MT-10**	**MT-10-R**
Single-task SAC	79.2%	66.5%
M-PPO [534]	25.0%	-
M-TRPO [535]	29.0%	-
TE [536]	30.0%	-
M-SAC [513]	53.3%	44.4%
M-M-SAC [513]	71.6%	57.7%
S-MRL-SM [537]	73.3%	**74.5%**
D-MRL-SM [537]	73.2%	67.9%
A-MFEA-RL (proposed)	**80.0%**	73.9%

Focusing first on single-task SAC, it attains competitive results in MT-10 that degrade when the RL tasks are not fixed (i.e. MT-10-R). This exposes that *SAC* requires to be trained over more frames to overcome its relatively low exploratory skills and the lack of knowledge transfer from other tasks. Furthermore, SAC requires a separate training process for every task, which incurs a significantly higher computational cost than A-MFEA-RL and other multitask approaches in the benchmark. Focusing on *MT-10*, the first three approaches (M-PPO, M-TRPO and TE) render average success rates notably below those yielded by M-SAC. Since the randomness of initial conditions imposed by *MT-10-R* makes the RL tasks harder to be solved, agents evolved for these three environments are expected to perform even worse. Therefore, M-PPO, M-TRPO and TE are left as baselines, and the discussion is centered on the last four schemes (M-SAC, M-M-SAC, S-MRL-SM and D-MRL-SM), which have been contributed to the community more recently than the former [513, 537].

This being said, A-MFEA-RL achieves the best average success rate (80%) for the fixed *MT-10* scenario, which throws evidence on its competitiveness even if, as stated before, the algorithm was unable to find good policies for the `pick-place` and `peg-insert-side` tasks. Indeed, the average score of A-MFEA-RL surpasses that of the second best approach in the benchmark by more than 6%. When the complexity of the scenario is increased by randomizing objects' and goal's positions (*MT-10-R*), S-MRL-SM outperforms the rest of state-of-the-art multitask learning counterparts. However, the average success rate of A-MFEA-RL gets very close (within a gap of less than 1%), a remarkable achievement given that A-MFEA-RL evolves significantly smaller networks ($|\mathbf{x}^p| \sim$ 87K parameters) than the ones in M-M-SAC ($\sim$ 327K), S-MRL-SM and D-MRL-SM ($\sim$ 270K).

Summarizing, experiments aimed at answering RQ2 have revealed that A-MFEA-RL elicits a competitive performance when compared to *avant-garde* multitask RL algorithms, both when the environments are fixed (*MT-10*) and when the goal is to train agents that generalize well when facing diverse environments (*MT-10-R*). As in the experiments for RQ1, synergies

Table 4.5: Success rates per task (%) obtained by the top three algorithms over MT-10, MT-10-R (RQ2), MT-50 and MT-50-R (RQ3). A stands for S-MRL-SM, B for D-MLR-SM and C for A-MFEA-RL. Complexity is denoted as E (*Easy*), M (*Medium*) and H (*Hard*).

Environment name (complexity)	**MT-10**			**MT-10-R**			**MT-50**			**MT-50-R**		
	A	B	C	A	B	C	A	B	C	A	B	C
`assembly` (H)	-	-	-	-	-	-	0	0	0	0	0	0
`basketball` (H)	-	-	-	-	-	-	0	0	0	22	33	0
`bin picking` (H)	-	-	-	-	-	-	0	0	0	0	0	11
`box close` (H)	-	-	-	-	-	-	44	44	0	22	33	0
`button press topdown` (M)	100	100	100	100	89	91	100	100	100	100	100	97
`button press topdown wall` (H)	-	-	-	-	-	-	67	78	100	67	100	100
`button press` (M)	-	-	-	-	-	-	44	67	100	44	55	100
`button press wall` (H)	-	-	-	-	-	-	100	100	100	100	100	98
`coffee button` (H)	-	-	-	-	-	-	44	78	100	56	89	100
`coffee pull` (M)	-	-	-	-	-	-	78	100	0	100	100	70
`coffee push` (M)	-	-	-	-	-	-	78	89	100	89	89	40
`dial turn` (H)	-	-	-	-	-	-	100	100	100	100	100	99
`disassemble` (H)	-	-	-	-	-	-	0	0	0	0	0	0
`door close` (H)	-	-	-	-	-	-	78	56	100	78	55	100
`door lock` (H)	-	-	-	-	-	-	89	100	100	89	89	100
`door open` (H)	100	33	100	100	100	100	78	67	100	67	67	100
`door unlock` (M)	-	-	-	-	-	-	78	89	100	89	100	100
`drawer close` (H)	100	100	100	100	100	100	79	89	100	67	78	100
`drawer open` (H)	0	33	100	33	0	99	22	33	100	22	44	98
`faucet close` (M)	-	-	-	-	-	-	100	67	100	78	44	81
`faucet open` (M)	-	-	-	-	-	-	89	89	100	89	67	91
`hammer` (H)	-	-	-	-	-	-	33	56	100	11	67	100
`hand insert` (M)	-	-	-	-	-	-	100	100	100	100	100	100
`handle press side` (H)	-	-	-	-	-	-	0	11	100	100	33	40
`handle press` (H)	-	-	-	-	-	-	89	78	60	100	78	35
`handle pull side` (H)	-	-	-	-	-	-	56	67	0	56	89	0
`handle pull` (H)	-	-	-	-	-	-	89	100	0	78	100	0
`lever pull` (M)	-	-	-	-	-	-	0	0	0	0	0	0
`peg insert side` (H)	67	33	0	56	56	0	0	22	0	44	33	0
`peg unplug side` (H)	-	-	-	-	-	-	100	100	0	100	100	0
`pick out of hole` (H)	-	-	-	-	-	-	0	0	0	0	0	0
`pick place` (H)	66	100	0	0	0	0	44	11	0	33	11	0
`pick place wall` (H)	-	-	-	-	-	-	44	33	0	33	0	10
`plate slide back side` (M)	-	-	-	-	-	-	100	89	40	78	89	45
`plate slide back` (M)	-	-	-	-	-	-	67	89	100	89	100	58
`plate slide side` (M)	-	-	-	-	-	-	100	89	100	55	100	100
`plate slide` (M)	-	-	-	-	-	-	33	100	100	78	78	77
`push back` (E)	-	-	-	-	-	-	89	100	0	89	100	71
`push` (E)	100	100	100	78	67	59	44	89	100	78	33	47
`push wall` (M)	-	-	-	-	-	-	56	33	100	55	44	47
`reach` (E)	100	100	100	100	100	91	100	100	100	100	100	98
`reach wall` (E)	-	-	-	-	-	-	100	100	100	100	100	98
`shelf place` (H)	-	-	-	-	-	-	0	0	0	44	55	0
`soccer` (E)	-	-	-	-	-	-	67	78	0	55	33	48
`stick pull` (H)	-	-	-	-	-	-	11	33	0	11	44	79
`stick push` (H)	-	-	-	-	-	-	0	0	0	11	0	100
`sweep into` (E)	-	-	-	-	-	-	100	78	100	67	89	80
`sweep` (E)	-	-	-	-	-	-	100	89	100	100	67	74
`window close` (H)	33	33	100	100	78	100	67	44	100	89	44	100
`window open` (H)	67	100	100	78	89	99	11	67	100	44	78	93
Average success rate	73.3	73.2	**80.0**	**74.5**	67.9	73.9	57.3	**62.0**	60.0	61.5	**62.1**	59.7

between tasks discovered by A-MFEA-RL have been found to remain in close accordance with those recently reported in [524]. In the next section this experimentation is scaled up to 50 simultaneously evolved tasks, towards ascertaining whether the competitive performance of A-MFEA-RL holds in more populated multitasking setups.

4.3.3. RQ3: Does A-MFEA-RL Scale Up Nicely When More RL Tasks are Simultaneously Evolved?

Throughout this subsection research question RQ3 is addressed, which aims to elucidate whether A-MFEA-RL contends with the rest of multitask RL methods when the number of simultaneously solved tasks K increases. To this end, the simulation setup is extended with the rest of the environments included in *Metaworld*, amounting up to $K = 50$ tasks, both with fixed (*MT-50*) and random (*MT-50-R*) initialization for each simulated episode. It is important to note that this extended set of experiments imposes significantly higher computational scales on A-MFEA-RL, which now operates on a population of $P = \sum_k P_k = 3000$ individuals evolved over 1000 generations.

Table 4.6: Success rate of tasks evolved for RQ3: MT-50 and MT-50-R.

Algorithm	**MT-50**	**MT-50-R**
M-PPO [534]	9.0%	-
M-TRPO [535]	22.9%	-
TE [536]	15.3%	-
M-SAC [513]	26.2%	25.7%
M-M-SAC [513]	28.6%	29.0%
S-MRL-SM [537]	57.3%	61.5%
D-MRL-SM [537]	**62.0%**	**62.1%**
A-MFEA-RL (proposed)	60.0%	59.7%

Concentrating the analysis on the success rates attained by all the methods in the benchmark, which are summarized in Table 4.6, from where it can be drawn that in *MT-50*, A-MFEA-RL achieves once again results close to the state of the art, outperforming M-PPO, M-TRPO, TE, M-SAC and M-M-SAC by a large margin. When turning the focus towards the more demanding *MT-50-R* scenario, the average success rate of A-MFEA-RL results to be more than double that of M-SAC and M-M-SAC. For both *MT-50* and *MT-50-R*, A-MFEA-RL gets very close to the success rates scored by S-MRL-SM and D-MRL-SM, which, as stated before, must be considered a good result considering that the evolved agents have approximately 70% less trainable parameters than the multitask RL models underneath S-MRL-SM and D-MRL-SM.

Rounding up the discussion, the focus is set on the per-task results obtained for these extended simulation setups, which are listed in Table 4.5. It can be noted that A-MFEA-RL is unable to solve any of the tasks that involve *grasping* or *picking* an object (e.g. `assembly` or `peg-insert-wide`), suggesting that further research efforts are needed for A-MFEA-RL to solve complex tasks, specially those that imply the concatenation of grasping and movement actions over time. Nevertheless, most of the remaining tasks are correctly solved, and the success rates achieved concur with the experiments conducted in RQ2. Therefore, it is fair to conclude that in terms of optimality, A-MFEA-RL scales up nicely when dealing with densely populated multitask scenarios. RQ5 will later revolve on whether this scalability can be also met in terms of computational cost.

4.3.4. RQ4: Does the Adaptive Knowledge Transfer of A-MFEA-RL Provide any Gain With Respect to Other Baseline Knowledge Transfer Methods?

This research question is devised to examine the contribution of the adaptive knowledge transfer performed in A-MFEA-RL. Specifically, A-MFEA-RL is compared to two different baselines: 1) random knowledge transfer (i.e. $rmp^{\ell}_{k,k'} = 1 \, \forall k, k'$) and 2) no knowledge transfer ($rmp^{\ell}_{k,k'} = 0 \, \forall k, k' : k \neq k'$). Moreover, an adaptation of the traditional MFEA has been added to the comparison, this adaptation has been tuned to utilize the same unified space than A-MFEA-RL. Therefore, the main differences between the adapted MFEA and aca-mfea-rl are the non-adaptive knowledge sharing and how operators are applied in the crossover stage (crossover + mutation in MFEA versus only layer-wise crossover in A-MFEA-RL). Results are reported from the best performance achieved by MFEA over a uniform 0.1-spaced grid of rmp values over the range $[0.1, 0.9]$. The selected value ($rmp = 0.3$) is close to those often encountered in works resorting to MFEA in multitask optimization settings, and is considered to provide with sufficient genetic knowledge exchange over the evolution [34]. The adaptive version of MFEA (MFEA-II) is not included in the comparison due to known shortcomings of the probability density estimation featured by MFEA-II in search spaces of large dimensionality.

Table 4.7: Comparing A-MFEA-RL to fixed RMP values and MFEA. The success rates for all tasks can be found in the git repository related to this project.

	MT-10-R	**MT-50-R**
No knowledge transfer ($rmp^{\ell}_{k,k'} = 0$)	73.2%	29.7%
Random knowledge transfer ($rmp^{\ell}_{k,k'} = 1$)	63.4%	35.3%
MFEA ($rmp = 0.3$) [34]	63.1%	26.2%
A-MFEA-RL (adaptive)	**73.9%**	**59.7%**

Table 4.7 summarizes the results of the experiments run for answering RQ4. In this table it can be first observed that in MT-10-R the performance of the adaptive approach is similar to that evolving all tasks in parallel without any knowledge transfer. However, the performance is more than 10% better when compared to its random transfer counterpart, showing the impact of negative knowledge transfer. This effect is greatly intensified in the *MT-50-R* scenario, where none of the baseline approaches achieves success rates close to those obtained by A-MFEA-RL. Here, two meaningful shortcomings of the non-adaptive approaches are drawn. On the one hand, when knowledge transfer is removed candidates are not allowed to be transferred among tasks, since their skill factor remains fixed over the search. There emerges

the importance of an adaptive version that allows evolved candidates to meet their most suitable task. On the other hand, if knowledge transfer is held at random, the evolutionary search is affected by negative transfers. In both cases the performance decays to nearly half of the success rate scored by A-MFEA-RL.

When it comes to A-MFEA-RL and MFEA, it is remarkable that the performance of MFEA is similar to that of random knowledge transfer, and lower than than of no knowledge transfer. Since MFEA utilizes the same unified search space of A-MFEA-RL, its differences with respect to *A-MFEA-RL* lie essentially on its operators. On the one hand, the thoughts on how the *Gaussian Mutation* operator affects evolution are corroborated. Applying the mutation operator after mating individuals waste the inherited knowledge generated during the crossover. On the other hand, the *simulated binary* crossover does not account for the layer-wise structure of the unified search space, hindering even further the convergence of the evolutionary search. This observation buttresses the need for an adapted search space, evolutionary operators and knowledge transfer that lie at the core of the proposed A-MFEA-RL.

4.3.5. RQ5: Is the Computational Cost of A-MFEA-RL Competitive with Respect to Other Multitask RL Methods?

The experiments end by delving into the analysis of the computational cost required by A-MFEA-RL to evolve the tasks under *MT-10-R*, *MT-50-R* and *single task* settings. A-MFEA-RL is compared to that of *M-SAC* [513], adapted so that an epoch in *M-SAC* is comparable to a generation in *A-MFEA-RL* in terms of the number of network updates and number of frames processed by one candidate at each generation. In terms of processed observations, at each epoch 300 episodes are used for evaluating every individual, yielding a total of $60 \cdot 10^3$ frames per candidate. Note that this configuration has been used exclusively for addressing RQ5 with an equal computational budget for both approaches. The discussion also considers *M-SAC* deployed on a Tesla V100 SXM2 GPU.

Table 4.8 collects the runtime statistics (average $\pm$ standard deviation) required to evolve one generation in A-MFEA-RL and to train one epoch in *M-SAC*. Statistics of *train_time* corresponds to the action of changing networks' parameter values. As such, in *A-MFEA-RL train_time* is defined as the time elapsed between the generation of new offspring, whereas in *M-SAC* it is the time taken for backpropagating gradients. On the other hand, *eval_time* is the time taken to evaluate the performance of the network. This entails evaluating the fitness function for the whole population in *A-MFEA-RL*, while in *M-SAC* the network is evaluated for $\mathsf{E} = 3$ episodes on each epoch, computing the success rate of the network as the mean of the last 100 epochs. As expected, both approaches require similar runtimes to complete an epoch: in A-MFEA-RL most of the time is consumed by the evaluation of the models,

Table 4.8: Average ± standard deviation of the runtime required by A-MFEA-RL, M-SAC to process the same amount of frames per epoch. Time reported in seconds and per epoch/generation.

	A-MFEA-RL		
	Single Task	MT10	MT50
train_time	3.6e-1±6.4e-2	4.2±7.4e-1	19.5±6.1e-1
eval_time	23.53±4.68	216.9±2.26	1036.3±8.7

	M-SAC		
	Single Task	MT10	MT50
train_time	34.82±4.27	389.53±7.5	1933.26±12.5
eval_time	2e-3±3e-4	5.23e-3 ± 8.9e-4	2e-1±3e-3

	M-SAC-GPU		
	Single Task	MT10	MT50
train_time	2.9±3.2e-2	42.78±11.76	301±16.28
eval_time	1.2e-3±1.3e-4	3.1e-3±4.7e-4	1e-2±7.8e-4

yet *training_time* is also significant and increases on a par with the number of simultaneously evolved tasks. Interestingly, dramatic runtime drops appear when *M-SAC* is run on GPU. This result engages with current trends on Evolutionary Computation aimed at implementing mmeta-heuristics on GPU processors [303], and stimulates future efforts towards GPU implementations of evolutionary multitask approaches to benefit from the computational efficiency of these massively parallel hardware architectures.

4.4. Conclusions

This chapter has elaborated on multitask RL, which aims at simultaneously learning to perform multiple tasks by exploiting the potential synergies existing among them. In this context, A-MFEA-RL has been presented, an evolutionary multitasking algorithm that has been adapted to deal with multitask RL environments via multifactorial evolutionary search principles. Its design has been shown to incorporate several key adaptations to allow for knowledge transfer among agents characterized by different neural network architectures. To begin with, the unified search space over which the evolutionary search is performed is designed in a layer-wise fashion, such that part of the genotype representing the weights and biases of every layer is shared across individuals. Furthermore, A-MFEA-RL also features an adaptive mechanism to update the probability that two individuals exchange knowledge contained in every layer along the search. These novel algorithmic ingredients give rise to an evolutionary multitasking approach that leverages positive relationships between tasks, and eludes negative intertask knowledge transfer.

These properties sought for A-MFEA-RL have been assessed over an extensive experimental setup comprising a maximum of 50 RL tasks. Results obtained over three different scenarios have been conclusive in regard to the capability of A-MFEA-RL to discover and exploit synergistic relationships between the tasks being solved. Furthermore, A-MFEA-RL has also been proven to scale smoothly with the number of evolved tasks, and to perform competitively when compared to other state of the art approaches for multitask RL, both in terms of the quality of the produced solutions (RL agents) and their computational cost.

5

ZERO-SHOT META-REINFORCEMENT LEARNING USING EVOLUTIONARY MULTITASK OPTIMIZATION

"Sometimes it's hard to tell the difference between determination and stubbornness, isn't it?"
- Celeste

5.1. Introduction

Up to this chapter, the Thesis has set clear that EM has attracted great efforts from the optimization community over recent years. This area has pioneered the exploitation of knowledge shared between related optimization problems. As stated in Chapter 3, there exists a noted lack of practical applications of EM. Some exceptions can be found in a few practical works, the most representative being [511], which uses EM to support the simultaneous training of several DL models. However, this general lack of practical applications of EM is a sign of the youth of this research area.

Throughout this Thesis, EM has been successfully applied to train multiple DL models simultaneously (Section 3.3.2), while their synergies are efficiently computed and leveraged to perform positive knowledge transfer between RL tasks. The approach introduced in Chapter 4 considers a large multitask learning scenario consisting of up to 50 tasks. Moreover, models evolved via a EM approach were found to perform on par with other DL-based approaches tailored to deal with multitask settings, evincing a promising future for this kind of approaches and the need for further research in this direction.

Despite the drawbacks of EDL when tackling large search spaces, population-based mmeta-heuristics have demonstrated their effectiveness at finding diverse and well-performing solutions, also taking advantage of previously evolved knowledge to boost the convergence of related problems over time (*dynamic optimization*). These traits can also be considered as a form of evolvability. Even if there is not a clear definition of this term, some works [539, 540] define *evolvability* as the time (or number of generations) an algorithm required to adapt itself to a change in the fitness landscape. This is a desirable skill for evolutionary meta-learning, where the fitness landscape is spanned by new modeling tasks that appear over time. In this envisaged scenario, evolutionary algorithms used to optimize and evolve models over time could provide the evolvability needed to retain and reuse knowledge among prevalent and newly emerging tasks.

The contribution made in this chapter introduces an extension of the proposal made in Chapter 4, so that the knowledge derived from a multitask configuration can be analyzed and reused to solve previously unseen RL tasks faster. A novel approach denoted as *Adaptive Meta-learning MFEA* (Adaptive Meta-Learning MFEA (AML-MFEA)) is designed to cope with this alternative scenario, incorporating the following modifications with respect to its predecessor:

- The module that computes task similarities is updated. Now, the so-called Centered Kernel Alignment (CKA) distance metric [331] is adopted to measure the similarity between evolved RL agents, in addition to the efficacy of historical crossovers previously considered. This endows the algorithm with the capacity to perform a two-fold weighted similarity search, historical and behavioral, taking in consideration the success rate of each task.

- Evolutionary meta-learning is approached by means of a bi-level optimization method that detects behaviorally similar tasks dynamically, and combine them into a new evolutionary process. This turns the approach into a bi-level optimization process similar to meta-learning, where the inner evolutionary loop tasks are jointly evolved, whereas in the outer loop the *meta-tasks*[1] are dynamically trained.

- A new experimentation setup evaluates the knowledge derived from the outer loop (i.e., *meta-tasks*) in newly arising tasks and for the most demanding meta-learning scenarios: *Few-Shot* [541] and *Zero-Shot* [542]. In *Few-Shot*, the model is granted a few generations to evolve the knowledge as per the feedback from the unseen tasks, while in *Zero-Shot* new tasks are evaluated directly, without any type of feedback nor adaptation whatsoever.

The rest of the chapter can be summarized as follows: in Section 5.2, works related to evolutionary multitasking are reviewed. Section 5.3 states the mathematical problem under scope, whereas Section 5.4 delves into the proposed approach, describing how the knowledge is consolidated, represented and adapted to *Zero-/Few-shot Learning*. In Section 5.5 the experimental setup and results are discussed. Finally, Section 5.6 elaborates on the conclusions drawn from this chapter.

1 In subsequent elaborations the term *meta-task* will be used to refer to the *average* RL task faced by the representative of a group of RL models with similar behaviors. In this regard, *meta-task* and the representative model of the group – namely, the *meta-model* – will be used indistinctly, as its meaning can be inferred from the context.

5.2. Related Work

Before describing the proposed AML-MFEA algorithm, we briefly revisit how MFEA can be adapted to evolve neural agents for RL (Section 5.2.1), the different ways in which meta-learning with neural networks has been studied so far (Section 5.2.2), and the few works that have been reported so far in the crossroads between evolutionary computation and meta-learning (Section 5.2.3).

5.2.1. Revisiting Adaptive MFEA for Reinforcement Learning

This chapter departs from the algorithm presented in Chapter 4, A-MFEA-RL. In this section A-MFEA-RL is briefly revisited in order to make the rest of the chapter easy to follow. Three are the cores empowering A-MFEA-RL: i) MFEA as the evolutionary engine performing the search in the multitask configuration; ii) the adaptive knowledge transfer module guided by the layer-wise rpm matrix; and iii) the crossover and mutation operators specifically designed to accommodate and exploit the layered neural architecture of the RL models.

The multitasking problem in A-MFEA-RL is tackled by means of MFEA (see Section 4.1.2), where a set of tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$ are evolved simultaneously. Each solution $\mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$ represents the weights and bias of the model assigned to a task $\mathcal{M} = \{M^{T_1}, M^{T_2}, \ldots, M^{T_K}\}$ encoded into the designed unified search space, which allows the approach to perform layer-wise TL automatically even if network structures are not identical. In order to encourage positive TL, synergies between each layer of every task are dynamically computed on each generation of A-MFEA-RL and stored as a matrix. This matrix consists of $rmp^{\ell}_{k,k'}$ entries for each task pair $k \neq k'$ and layer l. Additionally, the SBX crossover and *polynomial mutation* operators are adapted to facilitate layer-wise automatic knowledge transfer between tasks. As a result, A-MFEA-RL showed similar levels of performance when compared to other traditional RL approaches facing the same complex RL tasks, in terms of success rate and complexity. Departing from these outcomes, we arrived at the conclusion that RL in multitask settings were a promising path to follow, flowing into the research presented in this chapter.

This brief recapitulation establishes the grounds of this chapter. In what follows a review of the field of meta-learning and evolutionary meta-learning is performed, framing this work within trends identified in these areas.

5.2.2. Meta-learning with Neural Networks

Meta-learning is a challenging paradigm in DL which is designed to learn from how different algorithms learn in order to leverage that knowledge and improve the learning performance of new tasks, typically by means of a bi-level optimization process. Meta-learning has been a very prolific research area recently, having being tackled from different perspectives; 1)

Optimization-based, 2) model-based and 3) metric-based. It has been also applied to multiple fields such as image recognition, unsupervised learning or RL. For an in-depth explanation of the concepts introduced before, the reader is referred to [543]. Besides, metalearning has been proven to be closely related to multitasking: in [544] an exhaustive theoretical examination is performed, concluding that fast adaptation and efficient training can be achieved simultaneously by means of the mixture of these paradigms. An interesting instance of multitasking meta-learning is presented in [545], where authors adopt Model Agnostic Meta-learning (MAML [546]), an optimization-based meta-learning approach, and train the meta-learner with different multitask combinations. Unlike other approaches, the tasks considered can be from different domains (i.e. regression or classification). More examples of multitask meta-learning are collected in [547].

One of the limitations of multitasking for RL resides in the lack of a quality benchmark comprising a variety of related and unrelated tasks, limitation inherited in multitask meta-learning. For the scope of this work, *Meta-World*, the benchmark introduced in [517] that was used in Chapter 4, is adopted. Meta-World is designed for multitask and meta- RL, comprising 50 robotic manipulation tasks created with the *Mujoco* physics simulator engine. Meta-World provide the baseline algorithms for multitask and meta-learning, being MAML, RL^2 [548] and Probabilistic Embeddings for Actor-critic meta-RL (PEARL) [549] the algorithms employed to conduct the experiments on meta-learning. In this work, Meta-World is adopted as framework due to the complexity of its tasks even for traditional meta-learning algorithms, and because of the dimensionality and close connection to real-world RL robotic applications.

5.2.3. Evolutionary Meta-learning

Since the rise of EDL, stimulated by parallel achievements in meta-learning, evolutionary meta-learning has been approached from very distinct perspectives. By definition, meta-learning can be considered as an optimization problem and therefore, either the inner or the outer loop are typically replaced by an evolutionary algorithm. For instance, in [550] authors use Evolution Strategy (ES) to perform a meta-search of the parameters of a loss function that is used to train RL agents faster than other policy gradient methods. Applications that make an out-of-the-box use of evolutionary meta-learning can be found in [551], where the proposed meta-learning approach is able to automatically prune the channels of complex DL models. Other approaches attempt at finding the best performing set of algorithms for a task [552], or at realizing them as an ensemble of simple learners [553].

Even if the applications of evolutionary meatalearning are diverse, when shifting the discussion onto those approaches optimizing RL models, a bunch of proposals can be found, all seeking to evolve agents that adapt quickly to new tasks. This is commonly approached by means of ES [505] as an alternative to train RL agents, due to its proficiency when handling and optimizing multiple solutions in parallel, its efficient exploration skills and its ability to find solutions prone to adaptation. Embracing the search for solutions with *evolvability* skills, the work in [554] presents a meta-learning approach that pursues to evolve behaviors that adapt quickly (i.e. within a few SGD training steps) to new tasks, which is conceptually similar to MAML.

Evolutionary meta-learning has also been applied to real-world problems. Authors in [555] train a walking hexpod robot by means of ES-MAML, replacing the outer loop of MAML by ES. The method is trained off-line using the *Minitaur* robot simulator. Experimental results show that the algorithm is able to adapt under noisy scenarios, which can be extrapolated to the real hexapod robot facing physical variations like different battery voltages or different type of defects on the legs. Also related to a continual adaptation of the agent's behavior is the contribution presented in [556]. There, a bipedal walker is trained in such a way that its legs enlarge during execution and the network needs to be adapted by one of the three configurations presented; pure RL, pure EA or RL+EA. Results show that training the network with the configuration of RL+EA renders more diverse and best performing agents, while using pure EAs the approach fails to learn good behavioral policies.

Unlike the articles discussed in this section, the approach introduced in this chapter is, to the best of our knowledge, the first work introducing a full gradient-free evolutionary multitask approach for evolutionary meta-learning (AML-MFEA). By means of MFEA, the multitasking scenario and the redesigned affinity module, AML-MFEA is able to evolve tasks simultaneously while their relationships are dynamically computed. Evolving all tasks together enables the exploration of a large set of diverse yet qualified solutions that can help target tasks escape from local optima. However, completely different solutions (i.e., neural networks) might lead to agents encoding similar behaviors. For this reason, the affinity module is modified so that it takes in account both the historical effectiveness of inter-task crossovers and the distance among the behaviors encoded in the solution vectors. Finally, successfully evolved tasks are considered as the base to create *meta-tasks*, guided by the behavioral distance metric coined as Centered Kernel Alignment (CKA [331]). The following sections delve into the technical aspects of AML-MFEA, performing a detailed description of all its components.

5.3. Problem Statement

In this section the problem of evolutionary multitask for evolutionary meta-learning is mathematically formulated. For this purpose we start similarly to Section 5.2.1, by defining K tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$ that are simultaneously evolved by an EM approach. A best encoded solution candidate $\mathbf{x}_k^{*(t)} \in \chi^U$ is assigned to each task, representing the parameters of the best performing model found so far for task k in generation t, yielding a set of best candidates $\mathcal{B}^{(t)} = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_k^*\}^{(t)}$. $\mathcal{B}$ is composed by the best solution of each task, and can be regarded as the knowledge base provided to the outer evolutionary process (i.e. meta-learning).

Therefore, some fundamental issues reside under this configuration: i) Does $\mathcal{B}$ contain *meta-knowledge*? If so, what are the conditions to detect it?; ii) How can this knowledge be extracted and preserved over t?; and iii) How can *meta-knowledge* be dynamically updated over t? To address these questions, AML-MFEA introduces the concept of *meta-task*. Tasks that are dynamically created from the synergies exposed in the multitask scenario and separately evolved with their own population P_n^M, episodes per task E^M and the aim of generating a set of N *meta-tasks* $\mathcal{J}^{(t)} = \{J_1, J_2, \ldots, J_N\}^{(t)}$. Each meta-task comprises the knowledge of different simple tasks, and provides an enhanced evolvability to make new tasks be solved and/or be updated faster. Additionally, each meta-task is composed by the information about the tasks that created the group in which it is going to be evaluated, as well as their best solution.

We now proceed by introducing and describing the proposed approach, whose modifications with respect to A-MFEA-RL aim to tackle the questions posed above.

5.4. Proposed Approach: AML-MFEA

This section is devised to describe all technical aspects of AML-MFEA and all the components that allow the algorithm scale to extract and build a set of *meta-knowledge* groups derived from the knowledge of the multitask scenario. Figure 5.1 provides a general overview of the algorithm.

We begin from the bottom of Figure 5.1. The K tasks comprising the multitask learning configuration are evolved simultaneously by means of MFEA by means of the adaptive knowledge sharing strategy described in Section 5.4.1. In order to create *meta-tasks* (Section 5.4.2), at each generation t, the best candidates $\{\mathbf{x}_k^*\}_{k=1}^K$ are filtered based on their success rate by a threshold ϕ, their behavioral synergies are computed and, if any group has to be created or updated, they are refreshed. In what follows, each part of the algorithm is thoroughly detailed.

5.4.1. Updated Adaptive Knowledge Transfer Strategy

One of the core elements of AML-MFEA is the adaptive strategy adopted in MFEA, which enables an effective knowledge share between tasks. Unlike the layer-wise RMP matrix adopted for A-MFEA-RL, and stimulated by the results gathered in A-MFEA-RL, the RMP matrix in computed in a different fashion. So, the values comprising the RMP matrix can be defined as scalar values $rmp_{k,k'} \in \mathbb{R}(0,1)$ with $k \neq k'$ where k and k' denotes the indexes of the tasks for which the similarity is being computed. For each generation, each value in the RMP matrix is updated in a two-fold way. On the one hand, the generational efficiency of the crossover operator is considered. On the other hand, a neural network behavioral similarity metric coined as CKA

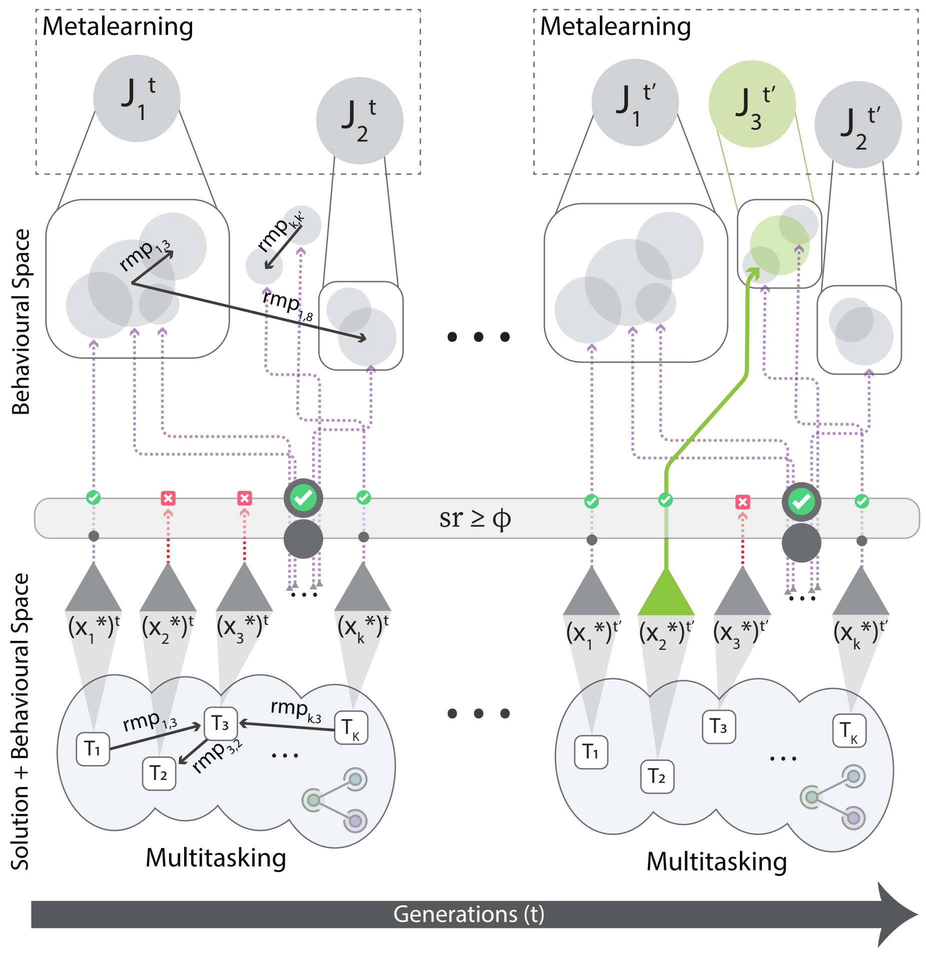

Figure 5.1: General overview of AML-MFEA general overview. In the middle of the diagram, the task filter ($sr \geq \phi$) separates the multitasking (bottom) and meta-learning (top) scenarios. Two different generations of the algorithm are represented, separated by ellipses. On the left, some of the best candidates in the multitask scenario do not meet the conditions to be input to the meta-learning scenario (red dashed lines), while other candidates fulfill them in the last generation (purple dashed lines), so it is not required to perform any *meta-task* update. On the right, a new task meets the condition (green line), therefore, it is taken into account in the *meta-task* creation process, from where $J_3^{t'}$ is dynamically created.

is implemented to find tasks that behave similarly as per this measure. This two-fold similarity measure transforms the old $rmp_{k,k'}$ into a weighted sum of these two factors, adopting the success rate as the weight balancing between them, and clipping the resulting value between rmp_{max} and rmp_{min} by the same `clip` function previously defined in Section 4.2.3. Mathematically:

$$rmp_{k,k'} = \texttt{clip}_{rmp_{min}}^{rmp_{max}} \left((1 - sr) \cdot rpm_{k,k'}^{hist} + sr \cdot rpm_{k,k'}^{cka} \right) \tag{5.1}$$

where sr denotes the success rate of the source task (i.e. the task with index k). Next, both mechanisms are detailed.

First, to take in consideration the historical efficiency of the crossover operator, two counters $C_{k,k'}^{+}$ and $C_{k,k'}^{-}$ are set to 0 at each generation. This resembles what was done in A-MFEA-RL except for the layered structure

(Section 4.2.3). These counters are used to record the times an offspring candidate's fitness is better (+) or worse (−) than that of its parent. Once all individuals are evaluated, the next formula is applied:

$$rmp^{hist}_{k,k} \leftarrow rmp^{hist}_{k,k} - \alpha \cdot C^{-}_{k,k'} + \beta \cdot C^{+}_{k,k'} \tag{5.2}$$

where $\alpha, \beta \in \mathbb{R}(0,1)$ are parameters used to control the effect of each counter in the $rmp^{hist}_{k,k'}$ update.

The reason to use a second factor to compute $rmp_{k,k'}$ is the need for finding and establishing connections between tasks that behave similarly, from which *meta-knowledge* can be derived. However, it is known that completely different network encodings can yield similar results and behaviors [557]. Therefore, either $rmp^{hist}_{k,k'}$ or any distance metric searching over the solution space $\mathcal{X}^U$ may not be a reliable way to test if models for two different tasks behave equally. Consequently, the exploration should be conducted over a *behavioral* search space.

For that purpose, the CKA similarity metric is adopted, CKA is able to measure the relationships between tasks using their representational matrices (i.e. the output of each layer of the neural hierarchy). However, it is of utmost importance to ensure that any pair of tasks to be compared have learned an acceptable representation of the environment in which they are trained (i.e., they feature a high success rate). Otherwise, CKA may not fulfill its purpose within the proposed algorithm due to the inexperience of the agents.

Therefore, defining a function $cka(\mathbf{x}^*_k, \mathbf{x}^*_{k'}, l)$ that computes the similarity of two - best - solutions corresponding to two different tasks (i.e. $k \neq k'$) and layer l, $rmp^{cka}_{k,k'}$ can be defined as:

$$rmp^{cka}_{k,k'} = \frac{1}{L_k} \sum_{l=1}^{L_k} cka(\mathbf{x}^*_k, \mathbf{x}^*_{k'}, l) \quad \forall \quad k, k' \in \mathbb{N}(1, K), \tag{5.3}$$

where L_k represents the number of layers of the network encoded in the solution vector $\mathbf{x}^*_k$. In [331] two versions of CKA are introduced, linear and RBF kernel. As stated in that work, both linear and kernel versions give similar outcomes in practice, hence, we adopt the same strategy as in [331] and use the linear version of CKA for AML-MFEA. Finally, it should be clarified that, mostly, $cka(\mathbf{x}^*_k, \mathbf{x}^*_{k'}, l) \neq cka(\mathbf{x}^*_{k'}, \mathbf{x}^*_k, l)$ mainly because of different network structures (i.e. $L_k \neq L_{k'}$), thus, k' is decoded to k and cka computed from the representational matrices of both tasks evaluated for the first task during some episodes. Additionally, evaluating network similarities over a set of episodes helps in the detection of similarities along any step within the episodes.

To finish, we note that, by means of the success rate sr, the $rmp_{k,k'}$ value balances smoothly between $rmp^{hist}_{k,k'}$ and $rmp^{cka}_{k,k'}$. This strategy allows the algorithm to automatically drift from exploring in the solution space considering the historical efficiency of inter-task crossovers ($sr \lessapprox 0.4$) to the exploitation of similar tasks in the behavioral space detected by means of CKA ($sr \gtrapprox 0.8$).

5.4.2. Meta-Knowledge Consolidation

We now describe the *meta-knowledge* consolidation process, from the conditions required for a task to take part in that process to the creation of *meta-tasks*. For the consolidation of *meta-knowledge* the basic piece of data provided to AML-MFEA is the best candidate set $\mathcal{B}^{(t)}$, the RMP matrix and the success rates of each task, all of them evaluated for generation t. The first step towards detecting different *meta-knowledge* groups is to exclude from the computation those models that do not perform well for their tasks. This is made by means of a threshold parameter $\phi \in \mathbb{N}(0,1)$, which represents the minimum sr value for a task to be considered to generalize acceptably to the environment they are aiming to solve.

Therefore, $\mathcal{B}^{(t)}_{skf}$ can be defined as the set of indexes of each task for which an individual in $\mathcal{B}^{(t)}$ encodes significant knowledge. We define any possible combination of tasks as $\mathcal{G}_n$, where n represents the order of the groups. For example, let $\mathcal{B}^{(t)}_{skf} = \{1,5,8\}$ be the indexes of the tasks where $sr \geq \phi$ in t, then, for $n = 3$ the resulting possible meta-tasks will be of order 3 and 2 (i.e. $n \leq |\mathcal{B}^{(t)}_{skf}|$), the group of order 3 is $\mathcal{G}_3 \equiv \{(1,5,8)\}$, while for $n = 2$ include all combination of length 2:

$$\mathcal{G}_2 \equiv \{(1,1),(1,5),(1,8),(5,1),(5,5),(5,8),(8,1),(8,5),(8,8)\}. \quad (5.4)$$

Once all possible *meta-tasks* of order n are computed, we check if all coupled values on each group meet the condition of $rmp^{cka}_{k,k'} \geq \gamma$. If so, a new meta-group is created. Let γ be a parameter that determines the amount of synergy required among tasks compounding the group $G_n \subseteq \mathcal{B}_{skf}$. By means of this parameter, the generalization difficulty can be tuned: when $\gamma << 1$, the algorithm will create bigger groups but less behaviors will be detected. Conversely, when $\gamma \approx 1$, the algorithm will only create groups of virtually repeated tasks. The process of meta-group creation is formalized in algorithm 2.

When a group is created, it is important to notice that the indexes conforming that group are excluded for the next loop. The reason is that it is preferable to create more groups comprising a great variety of tasks and better generalization skills than small groups compounded by behaviorally similar models.

5.4.3. Meta-Group Training and Evaluation

After following the process described in the last section to create the set $\mathcal{J}^{(t)}$ of *meta-tasks*, we proceed by explaining the process to train them, and how the selection of best individuals is performed. This section delves into the relevant aspects that make possible to learn agents with good generalization skills. First of all, it must be clarified that the network structure for each *meta-task* is predefined and made equal for all of them (as is specified in Section 5.5).

The training process of the evolutionary meta-learning layer proposed in this chapter can be directly compared to the outer loop of many traditional

Algorithm 2: Meta-group creation process.

```
1  Input: $\mathcal{B}^{(t)}_{skf}$, the set of groups containing meta-task indexes
2  Initialize meta-tasks as $\mathcal{J}^{(t)} \leftarrow \emptyset$
3  Set $n = |\mathcal{B}^{(t)}_{skf}|$
4  while $n \geq 2$ or $|\mathcal{B}^{(t)}_{skf}| \geq 2$ do
5      Compute groups of order $n \rightarrow \mathcal{G}_n$
6      for each group $G \in \mathcal{G}_n$ do
7          if if $rmp_{k,k'} \geq \gamma$ for each coupled $(k, k') \in G$ then
8              Create a new meta-group
9              Update $\mathcal{J}^{(t)} \leftarrow \mathcal{J}^{(t)} \cup G$
10             Delete indexes in $G$ from $\mathcal{B}^{(t)}_{skf}$ for next iteration
11     end
12     $n \leftarrow n - 1$
13 end
14 Return: $\mathcal{J}^{(t)}$ (the set of updated meta-tasks)
```

meta-learning approaches. However, unlike those methods, our purpose is not to construct a single model with superior generalization skills. By contrast, AML-MFEA aims to automatically detect those tasks that exhibit similar behaviors, so as to identify as many *meta-models* (see Footnote 1 in this chapter) as distinct behaviors are detected in the inner loop (multitasking), constructing a database of generalized behavioral patterns.

Every time a new *meta-task* is created a new evolutionary process is started. Each group is evolved separately, this is because each group is intended to encode different behaviors and thus, sharing knowledge between them should be avoided. Therefore, each task is in charge of evolving their own population, with its size and network architecture being defined beforehand. *Meta-tasks* are evaluated using the same operators as the multitask approach: SBX crossover, *Polynomial Mutation* and *Elitist Selection*. However, even if the fitness of meta-solutions is computed as in Section 4.2.4, here each *meta-task* is evaluated on each of the tasks assigned to them. This yields a fitness value for each of the environments $\mathbf{f}_J = \{f(\mathbf{x}_J, k)\} \forall k \in J$, being J a *meta-task* sampled from $\mathcal{J}^{(t)}$, and $\mathbf{x}_J$ an encoded solution belonging to the population P_J of that *meta-task*. Then, when all individuals in J are evaluated, they are ranked for each task in which they have been evaluated, and then ordered by the sum of their ranks (lower is better). However, matches may occur with this approach. To resolve them, the standard deviation of their rank positions is computed as a match breaking criterion. Consequently, the winner will be the solution that provides a more consistent generalization across all tasks.

In order to alleviate the computational cost derived from evaluating all tasks for each *meta-task*, the population is initialized with the best individuals of each task in the *meta-task* $\{\mathbf{x}^*_k \forall k \in \mathbb{N}(0, K) : k \in J\}$. With this simple strategy, *meta-tasks* are provided with the best valuable knowledge derived from the multitasking environment up to that generation.

5.4.4. Inference in Zero-Shot and Few-Shot Learning

After describing the extraction of knowledge and training of *meta-tasks*, we proceed by examining how the archive of *meta-knowledge* ($\mathcal{J}^{(t)}$) faces new tasks for the two challenging scenarios presented before: zero-shot and few-shot learning. Although they are widely used techniques, there are some considerations in AML-MFEA that should be emphasized. In AML-MFEA each *meta-task/meta-model* J_n can be treated as an expert within a mixture of experts ensemble [558]. Each expert has an assigned weight $\mathbf{W}$, where $\mathbf{W} = \{W_1, W_2, \ldots, W_n\}$, with n denoting the number of *meta-tasks* in $\mathcal{J}^{(t)}$. These parameters are in charge of guiding the contribution of each expert to the resulting aggregated action, and they are dynamically computed over episodes (weighted zero-shot) or set fixed (zero-shot). At inference time, each expert is fed with the same input (the observation of the new unseen environment). Figure 5.2 depicts a diagram describing how zero and few-shot learning are approached by AML-MFEA.

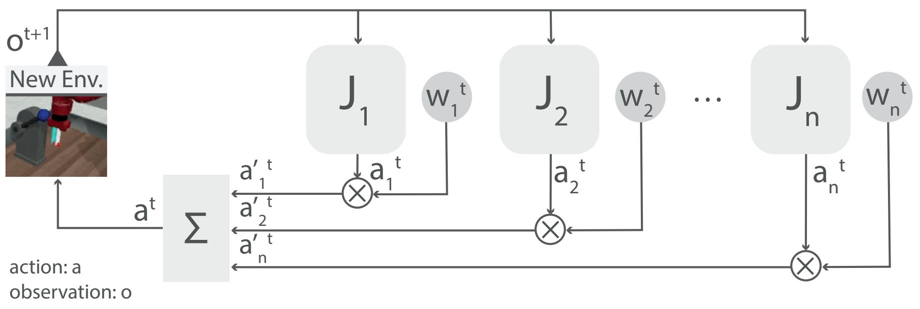

Figure 5.2: Diagram showing how inference is done by AML-MFEA when a new RL task arrives. The new environment provides the observation that each expert/model processes and evaluates. The action issued by each expert a_n^t is then weighted to yield $a_n^{',t}$ depending of the configuration (zero or few-shot). Finally, the action returned to the new environment is computed (a^t) as the mean of each component in $a_n^{',t}$.

Following Figure 5.2, three configurations are designed:

- *Zero-Shot learning*: new tasks are evaluated for λ episodes using only the knowledge in the set of *meta-tasks* (i.e. $\mathbf{W} = \{1\}^n$). In this configuration no parameters are updated, nor *meta-tasks* nor experts' weights.
- *Weighted Zero-Shot learning*: in this configuration the weights $\mathbf{W}$ of the ensemble are updated over a set of evaluated episodes λ. The update is performed at the end of each episode considering the mean reward of each expert. The parameters of the *meta-tasks* are not updated (i.e., they are kept fixed to their values evolved during training).
- *Few-Shot learning*: in this last configuration, AML-MFEA runs for some generations κ. We note that unlike in the other two configurations, in few-shot learning AML-MFEA evolves a new solution assigned to the new task. The knowledge in $\mathcal{J}^{(t)}$ is leveraged by using it in the initialization of the

population. Even if the algorithm runs for some generations, *meta-tasks* are not updated.

5.5. Experimental Setup & Results

By virtue of its design, AML-MFEA is able to dynamically detect task similarities at two levels, namely, over solution and behavioral spaces. This allows the multitask approach to generate valuable knowledge that is gathered and updated on-line to create *meta-groups*, as a basis to perform meta-learning. This section presents and discuss the results of a experimental setup designed to answer three *research questions*:

- RQ1: Is the knowledge consolidation performed by AML-MFEA robust?
- RQ2: Is AML-MFEA competitive in zero-shot and few-shot learning when compared to traditional meta-learning approaches?
- RQ3: Is AML-MFEA computationally efficient?

In order to assess the performance of AML-MFEA the MuJoCo physics simulator based *Metaworld* benchmark is used. Recalling the description from Section 4.3, *MetaWorld* is a benchmark for multitask learning and meta learning that provides 50 tasks that share some degree os similarities with others, however, they guarantee to be different enough to test meta-learning and multitasking approaches. As in A-MFEA-RL, the evaluation of the tasks is performed as the mean reward and success rate for $\mathsf{E} = 300$ episodes. Also in *MetaWorld*, authors provide several baselines against which to contrast new approaches, which are adopted for our experiments:

- RL^2 [548], which uses a slow RL algorithm to guide the training process of a fast RL algorithm. The former operates at a slower time-scale, producing agent policies that act as a supervisor for its fast counterpart. By contrast, the fast RL algorithm refreshes its learned policy in real time based on the feedback from the environment, hence quickly adapting to environmental changes and allowing for the exploration of new policies. As a result, RL^2 benefits from the stability and robustness of a slow RL approach with the agility and adaptability of a fast learning technique.
- Model Agnostic Meta-learning [546], which uses gradient-based optimization to learn a good initialization of the model parameters that can be quickly adapted to newly emerging tasks. The underlying idea behind MAML is to find a model that performs well on a wide range of tasks. This is accomplished by computing and using the gradient updates for each task to fine-tune the model for a new task.
- Probabilistic Embeddings for Actor-critic meta-RL [549], which relies on an actor-critic agent architecture. PEARL learns a mapping from the space spanned by observations to a lower-dimensional probabilistic embedding

that is suitable for policy learning. To this end, a variational objective is formulated to balance between information preserving in dimension reduction process, and the utility of the embedding space to learn new policies.

Following the categorization made in Section 4.3, tasks are divided as per their complexity (`EASY`, `MEDIUM` and `HARD`). Their architectures are kept to the same as the ones presented in Section 4.3. However, a new network architecture is defined for the meta-learning layer in AML-MFEA and due to the definition of *meta-tasks*. This is not trivial, as an oversized network can learn to separate all the tasks in the *meta-task* and overfit them, providing less generalization capability for the overall approach. Conversely, an undersized network could be unable to condense all the information encoded on each task in the *meta-tasks*, yielding bad outcomes. Thus, considering that tasks labeled as `HARD` are the ones that solve more complex tasks, and in consecuence are composed by more diverse behaviors (e.g., `reach`, `push`, `press a button`), the chosen neural network architecture is the one designed to encode `HARD` tasks. Thus, the composition and size of networks assigned to *meta-tasks* is that of the agent that learns to solve `HARD` environments. The list of network architectures is detailed in Table 5.1.

Table 5.1: Network architectures for every task complexity level. All layers are fully connected, comprising the indicated number of neurons (+ denotes layer concatenation).

Complexity	**Neural architecture**	**Parameters**
`EASY`	256 + 128 + 128 + output layer (S: Small)	$\sim$ 54K
`MEDIUM`	256 + 128 + 128 + 128 + output layer (M: Medium)	$\sim$ 70K
`HARD`	256 + 128 + 128 + 128 + 128 + output layer (B: Big)	$\sim$ 87K
Meta-task	256 + 128 + 128 + 128 + 128 + output layer	$\sim$ 87K

In our experiments, the whole set of parameters to be defined at the multitask level are enumerated in Table 5.2, together with all the new parameters introduced for AML-MFEA.

Table 5.2: Parameters used in the experiments.

Parameter	**Value**	**Parameter**	**Value**
P_k	60 for all tasks (i.e. $P = 60K$)	rmp_{min}	0.15
`E`	300 episodes	rmp_{max}	0.95
maxGen	600 (RQ1), 1000 (RQ2, RQ3)	α	0.01
start_rmp	0.80	β	0.10
P_n^M	$2 \cdot \lvert\mathcal{B}_{skf}^{(t)}\rvert$	$\mathtt{E}^M$	300
λ	60	κ	60
γ	0.8	ϕ	0.5
starting **W**	$\{1\}^n$		

Regarding the population of each *meta-task* P_n^M, it is configured to guarantee that *meta-tasks* are able to evolve with the lowest additional computational budget. It was empirically found out that twice the size of the formed *meta-task* group is a good approximation. To run the experimentation, a server equipped with four Intel Xeon Gold 5120 2.2 GHz processors and 1 Tb of RAM was used. Python source code implementing AML-MFEA and reproducing the results reported in what follows has been made available in https://gitlab.com/Ztira/aml-mfea for the sake of transparency and to support follow-up research.

5.5.1. Preliminary Analysis of Multitasking Results

From our experience with *MetaWorld* we already know that there are tasks that AML-MFEA will not be able to solve. For that reason, the amount of tasks with which it is trained is 32, setting aside 5 tasks for testing. For this experimentation AML-MFEA runs over $1,000$ generations with the configuration provided in Table 5.2. Table 5.3 presents the results of the multitasking scenario ($\mathcal{B}^{(t)}$), which exposes the basic knowledge required by AML-MFEA to perform meta-learning.

Table 5.3: List of the environments evolved by AML-MFEA, achieved success rate after 1000 generations and network architecture complexity.

Environment/task name	Success rate	Network size
`button-press-topdown`	0.92	Medium
`button-press-topdown-wall`	0.92	Big
`button-press`	1.0	Medium
`button-press-wall`	1.0	Big
`coffee-button`	1.0	Big
`coffee-push`	0.53	Medium
`dial-turn`	1.0	Big
`door-close`	0.95	Big
`door-open`	0.93	Big
`drawer-close`	1.0	Big
`drawer-open`	1.0	Big
`faucet-open`	1.0	Medium
`faucet-close`	1.0	Medium
`hammer`	1.0	Big
`handle-press-side`	0.98	Big
`handle-press`	0.82	Big
`handle-pull-side`	0.84	Big
`handle-pull`	0.98	Big
`lever-pull`	0.2	Medium
`reach`	0.88	Small
`push-back`	0.34	Small
`push`	0.29	Small
`plate-slide`	0.95	Medium
`plate-slide-side`	1.0	Medium
`plate-slide-back-side`	1.0	Medium
`peg-unplug-side`	0.35	Big
`soccer`	0.58	Small
`push-wall`	0.49	Medium
`reach-wall`	0.90	Small
`sweep-into`	0.56	Small
`window-open`	1.0	Big
`window-close`	1.0	Big

If one analyzes the success rates of each tasks, it is straightforward to notice that it is very low for some tasks, and will not contribute positively to the *meta-task* creation procedure (i.e., `lever-pull`, `push-back`, `push` and `peg-unplug-side`). On the contrary, others will narrowly pass the filter $sr \geq 0.5$ (namely, `coffee-push`, `soccer` and `sweep-into`). These environments will give clear insights about the internals of AML-MFEA in the next section.

5.5.2. RQ1: Is the knowledge consolidation performed by AML-MFEA robust?

This first research question checks whether the knowledge learned by the proposed meta-learning approach is robust, and if the groups embodied in the discovered *meta-tasks* qualitatively conform to intuition. To do this, two experiments are conducted. The first one is a "toy" experiment comprising 6 repeated environments (`window-open`, `window-close`, `faucet-open`, `faucet-close`, `button-press` and `button-press-wall`), amounting up to 12 tasks. In the second experiment, AML-MFEA faces a more realistic scenario covering all the environments detailed in Table 5.3.

Table 5.4: Meta-tasks derived from the environments evolved in the "toy-example" experiment. In brackets the complexity of each network architecture is represented as S (small), M (medium), and B (Big).

	Env. Name (Complexity)	**Meta Success Rate** (meta-model)	**Base Success Rate** (individual models)
Group 1	`window-close` (Big)	1.0	1.0
	`window-close-2` (Big)	1.0	1.0
Group 2	`faucet-close` (Medium)	0.93	1.0
	`faucet-close-2` (Medium)	0.93	1.0
	`faucet-open` (Medium)	1.0	1.0
	`faucet-open-2` (Medium)	1.0	1.0
	`button-press-wall` (Big)	1.0	1.0
	`button-press-wall-2` (Big)	1.0	1.0
Group 3	`window-open` (Big)	1.0	1.0
	`window-open-2` (Big)	1.0	1.0

Results shown in Table 5.4 demonstrates that AML-MFEA is able to correctly detect coupled tasks and cluster them in groups containing, at least, those twin tasks. It is remarkable that the synergies between the `button-press` environments are not detected, which suggests that both environments are solved by slightly different behaviors. In fact, a visual exploration of how the two evolved agents solve this environment supports this statement: the agents solve the task in different behavioral ways: one proceeds straight ahead, whereas the other by executing a quick side-to-side movement. Similarly, the three tasks gathered in *Group 2* seem counterintuitive, since `faucet-open` and `faucet-close` can be thought to require agents that behave in an opposite way. However, once again a quick visual exploration of the behavior of both evolved agents clarifies this surprising result: `Faucet-close` is solved by the same behavior, but not touching the handle of the faucet, but its back part. This is illustrated in Figure 5.3.

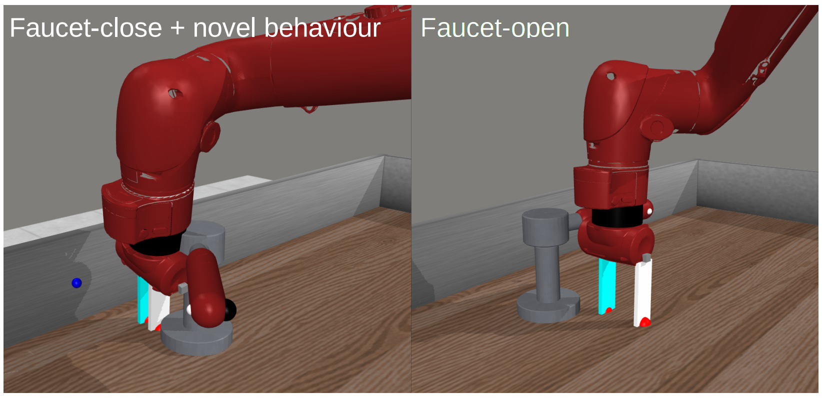

Figure 5.3: Snapshots showing how `faucet-open` and `faucet-close` are jointly solved by the discovered *meta-task*. To close the faucet, the meta-model learns to solve it in an innovative way.

In what refers to the second more realistic experiment, Table 5.5 presents the *meta-groups* learned over the entire set of environments under consideration. There, seven groups built from the results of the multitask scenario are portrayed. In order to make a deeper analysis of the nature of the tasks, and how it affects the *meta-tasks'* efficacy, the success rate of each task in every *meta-task* is added.

Table 5.5: Groups created from the results in Table 5.3. In brackets the complexity of each environment is represented as $S = Small$, $M = Medium$ and $B = Big$.

	Env. Name (Complexity)	**Meta Success Rate** (meta-model)	**Base Success Rate** (individual models)
Group 1	`button-press-wall` (B)	0.95	**1.0**
	`window-open` (B)	1.0	1.0
Group 2	`button-press` (M)	0.92	**1.0**
	`handle-press` (B)	**0.83**	0.82
Group 3	`reach` (S)	**0.94**	0.88
	`reach-wall` (S)	**1.0**	0.9
Group 4	`button-press-topdown` (M)	**0.94**	0.92
	`button-press-topdown-wall` (B)	**0.94**	0.92
	`hammer` (B)	0.0	**1.0**
Group 5	`soccer` (S)	0.2	**0.58**
	`sweep-into` (S)	**0.57**	0.56
Group 6	`drawer-open` (B)	0.0	**1.0**
	`faucet-open` (M)	0.94	**1.0**
	`plate-slide` (M)	0.95	0.95
	`plate-slide-side` (M)	0.0	**1.0**
Group 7	`coffee-push` (M)	0.23	**0.53**
	`plate-slide-back-side` (M)	0.22	**1.0**

Results in Table 5.5 lead to interesting, and coherent insights on the importance of a valuable knowledge basis to perform this type of evolutionary meta-learning. We start the discussion with *Group 1*, compounded by `button-press-wall` and `window-open`. A glance at how tasks are solved[2]

2 Videos of agents solving the tasks can be visualized in https://meta-world.github.io/

even if the end of the episodes for both environments is quite different from each other. The start of the episode and until `button-press-wall` completes, both are exceptionally similar. Thus, it can be intuitively thought that `window-open` *contains* `button-press-wall` within its behavioral space (i.e., their joint cka value is close to 1). Additionally, it can be observed that two models with a large architectural size can be merged into a single model of the same dimensions (i.e., a model with half the number of parameters) that performs almost equally.

A similar reasoning can be done in what refers to *Group 2*, where `button-press` and `handle-press` are jointly evolved. Here, results show that both environments can be grouped successfully. However, as in the first experiment conducted in this section, a visual analysis of the behavior of their evolved agents reveal that they solve them using the same template. Another remarkable fact is that for this experiment, as in the previous "toy" experiment, `button-press` and `button-press-wall` are assigned to different groups even if they can be very similar to each other. This evinces that the algorithm is stable for different scenarios.

We now focus on *Group 3* and *Group 4*. In both cases, the environments grouped as similar as per the similarity of their evolved agents are the version with and without wall of `reach` and `button-press-topdown`. This again echoes the effectiveness of the proposed adaptive similarity metric (Section 5.4.1) when finding tasks that behave similarly. However, while *Group 3* is able to perform even better than their detached alternatives, in *Group 4* the information of `hammer` is absolutely lost. A visual analysis of how hammer is solved by its evolved agent displays that, even if the agent reaches maximum success rate, it solves the environment by exploiting some design failures of the environment itself, rather than by following the expected behavior.

Some examples of the negative effect that an under-evolved task can induce in the *meta-task* can be noticed in *Group 5* and *Group 7*. In the former, tasks `soccer` and `sweep-into` seem to be visually similar to each other. Unfortunately, the *meta-task* is unable to merge both models into a single one. Ending with *Group 6*, `faucet-open` and `plate-slide` are clearly behaviorally similar. However, `drawer-open` and `plate-slide-side` only show similarities for some early steps of the episode. This limitation is introduced by the computation of cka and its dependency to different episode lengths. To be more specific, this effect is clearly visible on `drawer-open`, where the way to the handle of the drawer is similar to the behavior in `faucet-open` or `plate-slide`, but the way back to opening the drawer is not mirrored in any of the other tasks. The reason why `plate-slide-side` is in *Group 6* is not trivial, and would need further analysis of AML-MFEA to derive the environmental artifacts producing this unexpected result.

Throughout this section, an assessment of the quality of created *meta-tasks* has been done. In general, the approach has been proven to expose good performance at finding behaviorally similar tasks. In the next section, the generalization skills of the agents involved in the *meta-tasks* is tested over the zero-shot and few-shot scenarios introduced previously.

5.5.3. RQ2: Is AML-MFEA competitive in zero-shot and few-shot learning when compared to traditional meta-learning approaches?

This section is devised to answer the question of how learned *meta-tasks* generalize to new tasks in some of the most complex meta-learning configurations: zero-shot and few-shot. Departing from the configurations introduced in Section 5.4.4, this section first compares zero-shot and weighted zero-shot learning to check if new tasks leverage the generated meta-knowledge (embodied in the meta-models of the meta-tasks) properly. Next, results over the weighted zero-shot and few-shot learning are contrasted to other state-of-the-art meta-learners (R^2, MAML and PEARL), all introduced in Section 5.2.3. To this end, two setups are considered: one designed ad-hoc based on the reduced set of tasks considered in Table 5.3, from which 5 tasks have been left out as *newly unseen* tasks; and a second one, following the distribution of training and test tasks imposed by the so-called ML-45 benchmark described in the seminal work of MetaWorld [517]. The purpose of the first experiment is to place value on the proposed algorithm with respect to a randomly initialized model focusing on zero-shot scenarios. The second experiment aims to ascertain whether these results are competitive with respect to avant-garde non-evolutionary alternatives.

Table 5.6: Results obtained for the first meta-learning scenario devised to answer RQ2.

Environment	**Zero-Shot**	**Weighted Zero-Shot**	**Random**
`window-open`	0.6	**1.0**	0.0
`button-press-topdown`	0.0	**0.91**	0.0
`dial-turn`	0.34	**0.45**	0.0
`door-open`	0.0	0.0	0.0
`drawer-open`	0.0	0.0	0.0

Outcomes in Table 5.6 show a clear distinction between the approaches considered, being weighted zero-shot the absolute winner and *random* the worst (considered the simplest baseline to perform zero-shot learning). However, for tasks `door-open` and `drawer-open`, similar results are obtained by zero-shot and *random*, giving an insight about the need for guiding new tasks by the weighted version of zero-shot AML-MFEA. Due to the design of *meta-tasks*, they will encode very varied behaviors. Thus, with the arrival of a new task, opposite behaviors will negatively affect the predicted action, leading the model to poor results.

Results for the second experiment related to RQ2 are shown in Table 5.7, which summarizes the success rates obtained by AML-MFEA and the rest of

Table 5.7: Results obtained over the ML-45 meta-learning scenario of MetaWorld. In the table, *n.r.* stands for *not reported in [517]*.

	RL^2		**PEARL**		**MAML**		**AML-MFEA**	
	Zero-shot	Few-shot	Zero-shot	Few-shot	Zero-shot	Few-shot	Zero-shot	Few-shot
`bin-picking`	n.r.	**0.11**	n.r.	0.004	n.r.	0.0	0.0	0.0
`door-unlock`	n.r.	0.57	n.r.	0.42	n.r.	**0.7**	0.022	0.21
`hand-insert`	n.r.	0.28	n.r.	**0.30**	n.r.	0.0	0.0	0.13
`door-lock`	n.r.	0.35	n.r.	0.32	n.r.	**0.9**	0.0	0.75
`box-close`	n.r.	**0.31**	n.r.	0.04	n.r.	0.0	0.01	0.1
Mean	0.02	**0.33**	0.01	0.22	**0.03**	0.32	0.006	0.24

meta-learning techniques over the test RL tasks defined in the ML-45 scenario of MetaWorld. Here, the weighted version of AML-MFEA is considered for the zero-shot scenario. In this table we first observe that the average success rate attained by all approaches in the comparison is very low in the zero-shot scenario, whereas allowing the models to update on a few samples delivered by the test environments (few-shot scenario) yields notable average success rate improvements. In more detail, the average score achieved by PEARL lags slightly behind that of AML-MFEA in the few-shot setting, as this algorithm fails to generalize properly when solving the `door-lock` task. Interestingly, MAML seems to outperform AML-MFEA on average, but an inspection of the success rates per task uncovers that MAML fails to solve 3 out of the 5 tasks. This variability suggests that the performance of MAML might degrade when facing other set of unseen tasks. In line with the statements in the MetaWorld original publication, RL^2 performs best on average in the few-shot setting, outperforming the rest of algorithms across all tasks except `door-lock`. For the zero-shot setup, only average scores were reported in [517]. Even so, the proposed AML-MFEA fails to perform competitively. This low performance can be due to different reasons, e.g. the lack of information to properly gauge the relevance that each of the meta-models should have when voting their predicted policies. More research is needed in this regard, which will be discussed in the closing part of this Thesis.

5.5.4. RQ3: Is AML-MFEA computationally efficient?

Meta-learning is, by nature, computationally demanding, mainly due to its bi-level training architecture (i.e inner loop and outer loop). The inner loop implies training on a wide set of tasks from which the outer loop learns, hence the computational cost of the inner loop depends of the number of tasks considered for training. For this reason, most of the contributions in the related literature do not report any results in terms of efficiency (training time). Moreover, it is remarkable that the time elapsed to train a meta-learning approach also depends on the available computational resources, the parallel implementation of agents, and the device on which the learning algorithms are deployed (CPU, GPU or TPU).

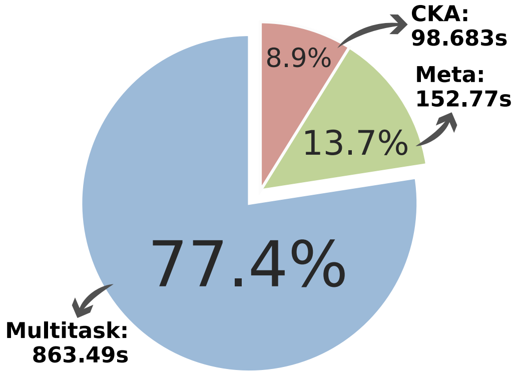

Figure 5.4: Distribution of the training time of AML-MFEA for one generation.

The implementation of AML-MFEA furnished in this Thesis does not run over any GPU/TPU processor. However, each of its agents runs in parallel. As a result, the computational cost of AML-MFEA decreases drastically, but still remains costly. As illustrated in Figure 5.4, 77.4% of the time AML-MFEA evaluates the tasks in the multitasking scenario, what amounts to almost 864 seconds per generation. The most costly operation in AML-MFEA is the evaluation of candidate solutions on each environment/task, since the multitask, cka and the meta-tasks computation involve evaluating candidates. Training *meta-tasks* consumes 13.7% of the total generation time (i.e., 152.77 seconds), whereas computing the cka measure of 32 tasks takes 99 seconds, which is a relevant amount that relates to the number of episodes used to measure the similarity of the networks. Even if the time elapsed to evolve multiple tasks at the same time cannot compete with traditional RL multitask algorithms running on GPU (see Table 4.8), AML-MFEA requires a minimal extra amount of time. Thus, the overall AML-MFEA could not be deemed computationally efficient with respect to other traditional multitask approaches, but the designed meta-learning module can be considered to be efficient as it is designed to be effective by performing meta-learning with few additional candidates. However, considering unlimited resources, this time will linearly increase by ≈ 1.3 seconds per evaluated candidate.

5.6. Conclusions and Research Directions

This chapter has presented AML-MFEA, an evolutionary meta-learning approach relying on evolutionary multitask optimization. AML-MFEA is able to simultaneously evolve multiple tasks whose synergies are dynamically computed. A similarity detection module between models evolved for every task allows AML-MFEA to detect similarities between tasks, not only based on the efficacy of inter-task crossovers, but also in what refers to the behavioral space, using the network similarity metric *CKA* for this latter purpose. By combining these two factors into a single metric, similar tasks can be detected and grouped into *meta-groups*. A different optimization process that imitates the outer loop of traditional meta-learning architectures is then performed to hold different network behaviors that could be used to train new tasks more efficiently.

To assess the performance of AML-MFEA, an extensive experimental setup is designed, comprising up to 32 tasks that are simultaneously evolved, leading to a total of 7 *meta-groups*. Throughout the experimentation the robustness of the consolidated knowledge has been tested. Results show that the discovered task groups not only meet human intuition regarding how two grouped tasks are solved, but also permit to identify alternative ways by which an agent can learn to solve a task by exploiting tweaks in the environments under consideration. The competitiveness of AML-MFEA in zero-shot and few-shot learning has been assessed, showing that it performs on average competitively with respect to other methods from the state of the art, yet still being far from the unrivaled generalization scores of tier-one methods. Finally, AML-MFEA has been analyzed from the perspective of computational efficiency viewpoint. Conclusions in this regard go in line with those of the previous chapter, namely, the overall computational complexity of AML-MFEA is notably higher than that of other approaches. However, the meta-learning processing layer added on top of the multitask layer does not add a significant computational penalty to the overall time cost of the proposed approach.

Part IV

FINAL REMARKS

6 FINAL REMARKS

"We can appreciate the entire journey by looking back at how far we have come."
– Antichamber

6.1. Conclusions

This Thesis has revolved around the combination of evolutionary computation and DL, which has become increasingly relevant in recent years due to the complexity of the optimization problems that can be formulated under this family of models. Indeed, evolutionary computation is comprised by a diverse landscape of algorithms capable of optimizing complex non-linear problems thanks to gradient-free search mechanisms, such as those underneath genetic algorithms. DL allows for powerful representations of complex data spaces through layered structures of artificial neurons and neural processing units. The fact that the topology selection, hyper-parametric tuning and parameter training of these networks are optimization problems has paved the way for evolutionary computation to address such problems efficiently. By combining these two areas, it is possible to optimize the architecture and parameters of DL models in both an efficient and effective manner. The research work condensed in this Thesis has aimed to shed light on this technological crossroads from a twofold perspective: i) a study of the related literature, stepping beyond a simple recollection of contributions to offer insights supported by side experiments and prospects motivated by the lessons learned therefrom; and ii) an exploration of other scenarios where evolutionary computation can make a difference in the adaptability and performance of DL models.

While combining Evolutionary Computation and Deep Learning holds significant promise, it is important to note its limitations and drawbacks. This has been the goal of Chapter 2, starting from a taxonomy of the optimization problems that have been solved in the literature with metaheuristic optimization algorithms. After the critical review of the literature following the defined taxonomy, several insights supported by experiments have been identified in EDL. The first is that, indeed, evolutionary computation can boost the search for optimal topology and hyper-parameter settings in modeling tasks approached by DL models. By contrast, current efforts to use evolutionary computation for trainable parameter optimization (model training) fall short when dealing with realistic network sizes. Experiments with evolutionary techniques suited to deal with large-scale optimization problems have been proven to perform subpar with respect to gradient backpropagation solvers. Based on these experimental observations and other caveats identified in the reviewed literature, Chapter 2 has finished by outlining several research directions of interest for the community, among which two have spurred the rest of the Thesis: i) the exploitation of problem-specific knowledge during the search; and ii) the potential of EM to foster the exchange of knowledge between different instances of problems related to DL.

Chapter 3 follows up the advocated potential of EM by inspecting what has been done in this recent subarea of evolutionary computation. EM refers to a type of optimization algorithms that evolve multiple solutions at the same time for a number of related optimization tasks. As opposed to traditional EAs, which focus on evolving a single solution for a single task, the process of evolving solutions for multiple tasks can lead to a more efficient and effective optimization process, as solutions that are suited to multiple tasks simultaneously can be explored. As per the thorough literature review in Chapter 3, EM has been applied to a variety of application scenarios, including machine learning. In this context, renowned success histories on the combination of evolutionary computation and RL have been motivational to explore the possibilities to leverage EM to realize and expedite the transfer of knowledge in RL setups comprising several tasks, as it occurs in multitask RL and meta-RL. RL algorithms train agents to make decisions in an environment by maximizing a reward signal, while Evolutionary Computation provides a means to optimize the agent's policy through evolutionary search. When the policy is produced by a DL model that maps observations to actions, the use of EM to evolve the trainable parameters of different models can realize an implicit knowledge transfer between such models, realizing a sort of evolutionary TL.

Chapter 4 has tackled the use of evolutionary multitask for multitask RL, which aims to train agents that perform multiple tasks simultaneously, where the goal is to leverage the knowledge gained from one task to improve performance on another task. TL has emerged as a promising approach in this context, allowing agents to share common knowledge and representations between tasks, thereby improving the overall performance and sample efficiency. By leveraging transfer learning, the agent can take advantage of the knowledge gained from one task to quickly adapt to new tasks, reducing the amount of training data required and speeding up the learning process. Specifically, the Thesis at this point has proposed an adaptive EM approach that estimates the amount of knowledge transferred indirectly between RL tasks. This estimation evolves during the evolutionary search, so that synergistic relationships can be tailored as per the convergence of the search and the maturity of the policies embodied in the evolved models. Experimentally it has been found out that when it comes to performance, the proposed algorithm can rival other modern non-evolutionary methods multitask RL. Unfortunately, the high computational cost of running an evolutionary search makes the proposed approach an unfeasible replacement for other methods whenever computational efficiency enters the picture as another criterion to be met.

Next, Chapter 5 has built upon its predecessor to address the meta-RL setup, where the goal is to train a set of behaviors (models) able to help agents to quickly adapt to new tasks with minimal data. TL has emerged as a promising approach in meta-RL, allowing the agent to leverage prior knowledge from related tasks to improve its performance in new environments. Additionally, TL can be used to improve the efficiency of the meta-learning process itself, allowing the agent to learn more effectively from past experiences. Chapter 5 has capitalized on this idea to assess whether the transfer of knowledge between training tasks efficiently realized by EM can be processed further to infer behavioral commonalities between the learned models, which can serve as a knowledge base to address newly arising tasks more effectively. In doing so, a measure of similarity between neural networks has been adopted to compare the evolved agent models and group them in meta-tasks, i.e., tasks whose evolved models behave similarly to each other as per their trainable parameters. The inference of such meta-tasks yields an ensemble of agents with which to solve new unseen tasks in a robust fashion, considering all typical behaviors of models learned for the training tasks. Experimental results have exposed, on one hand, that the composition of meta-tasks conform to intuition, successfully grouping tasks whose solutions (paths) can be thought to be related to each other. By contrast, in zero- and few-shot learning scenarios the proposed approach performs competitively with respect to other non-evolutionary approaches, but is still far from the generalization performance of the best known method in this area (RL^2).

An overarching conclusion drawn from the findings reported in the Thesis is that EDL is an area full of lights and shadows. Light emanates from the automated construction of DL architectures (Auto-ML) and hyper-parameter tuning, as the Thesis has clearly exposed. Unfortunately, end-to-end evolutionary training of realistically sized DL models cannot compete with gradient backpropagation approaches. However, recent developments in the use of evolutionary computation to construct models based on processing primitives (AutoML-zero [559]), continual/lifelong machine learning [560] or the diversification of behaviors for evolved models (Quality-diversity optimization [561]) have reignited the interest in the adaptability of machine learning models through evolution. This Thesis has advanced in this direction, proving that machines can autonomously improve their learning process by exploiting synergies between tasks through evolution, reducing the data required for the purpose and producing models prepared to cope with unseen tasks. Multitasking and generalization towards the unknown, together with other properties such as multi-modality and knowledge representation, persistence and retrieval, are within the desiderata of General Purpose Artificial Intelligence (GPAI). This Thesis and contributions in years to come will surely showcase the crucial role expected for evolutionary computation in the design of models addressing this paradigm.

6.2. List of Publications

The research conducted while pursuing this Thesis has given rise to several publications in conferences and journals, which are listed below:

- **Journal publications**:
  - Aritz D. Martinez, Javier Del Ser, Esther Villar-Rodriguez, Eneko Osaba, Javier Poyatos, Siham Tabik, Daniel Molina, Francisco Herrera, “Lights and shadows in Evolutionary Deep Learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges”, Information Fusion, Vol. 67, pp. 161–194, 2021. JCR: 17.564 (ranking: 4/145, Q1, COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE).
  - Aritz D. Martinez, Javier Del Ser, Eneko Osaba, Francisco Herrera, “Adaptive multifactorial evolutionary optimization for multitask reinforcement learning”, IEEE Transactions on Evolutionary Computation, Vol. 26, N. 2, pp. 233–247, 2021. JCR: 16.497 (ranking: 5/145, Q1, COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE).
  - Eneko Osaba, Javier Del Ser, Aritz D. Martinez, Amir Hussain, “Evolutionary multitask optimization: a methodological overview, challenges, and future research directions”, Cognitive Computation, Vol. 14, N. 3, pp. 927–954, 2022. JCR: 4.890 (ranking: 49/145, Q2, COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE).
- **Contributions to conferences**:
  - Aritz D. Martinez, Eneko Osaba, Javier Del Ser, Francisco Herrera, “Simultaneously evolving deep reinforcement learning models using multifactorial optimization”, IEEE Congress on Evolutionary Computation (CEC), pp. 1–8, 2020.

In addition to the publications related to the Thesis, the author has also actively collaborated in other research works, leading to the following publications:

- **Journal publications**:
  - Javier Poyatos, Daniel Molina, Aritz D. Martinez, Javier Del Ser, Francisco Herrera, “EvoPruneDeepTL: An evolutionary pruning model for transfer learning based deep neural networks”, Neural Networks, Vol. 158, pp. 59–82, 2023. JCR: 9.657 (ranking: 16/145, Q1, COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE).

  - Eneko Osaba, Javier Del Ser, Aritz D. Martinez, Jesus L Lobo, Francisco Herrera, "AT-MFCGA: An adaptive transfer-guided multifactorial cellular genetic algorithm for evolutionary multitasking", Information Sciences, Vol. 570, pp. 577–598, 2021. JCR: 8.233 (ranking: 16/164, Q1, COMPUTER SCIENCE, INFORMATION SYSTEMS).
  - Aritz D. Martinez, Eneko Osaba, Miren Nekane Bilbao, Javier Del Ser, "Let nature decide its nature: On the design of collaborative hyperheuristics for decentralized ephemeral environments", Future Generation Computer Systems, Vol. 88, pp. 792–805, 2018. JCR: 5.768 (ranking: 8/105, Q1, COMPUTER SCIENCE, THEORY & METHODS).

- **Contributions to conferences**:
  - Eneko Osaba, Javier Del Ser, Aritz D. Martinez, Jesus L Lobo, Antonio J Nebro, Xin-She Yang, "MO-MFCGA: Multiobjective multifactorial cellular genetic algorithm for evolutionary multitasking", IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8, 2021.
  - Eneko Osaba, Javier Del Ser, Aritz D. Martinez, Jesus L Lobo, "A multifactorial cellular genetic algorithm for multimodal multitask optimization", IEEE Congress on Evolutionary Computation (CEC), pp. 1–8, 2022.
  - Alain Andres, Esther Villar-Rodriguez, Aritz D. Martinez, Javier Del Ser, "Collaborative exploration and reinforcement learning between heterogeneously skilled agents in environments with sparse rewards", International Joint Conference on Neural Networks (IJCNN), pp. 1–10, 2021.
  - Javier Del Ser, Eneko Osaba, Aritz D. Martinez, Miren Nekane Bilbao, Javier Poyatos, Daniel Molina, Francisco Herrera, "More is not always better: insights from a massive comparison of meta-heuristic algorithms over real-parameter optimization problems", IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–7, 2021.
  - Eneko Osaba, Aritz D. Martinez, Akemi Galvez, Andres Iglesias, Javier Del Ser, "dMFEA-II: An adaptive multifactorial evolutionary algorithm for permutation-based discrete optimization problems", Genetic and Evolutionary Computation Conference Companion, pp. 1690–1696, 2020.
  - Eneko Osaba, Aritz D. Martinez, Jesus L Lobo, Javier Del Ser, Francisco Herrera, "Multifactorial cellular genetic algorithm (MFCGA): Algorithmic design, performance comparison and genetic transferability analysis", IEEE Congress on Evolutionary Computation (CEC), pp. 1–8, 2020 (finalist, Best Paper Award)
  - Aritz D. Martinez, Eneko Osaba, Izaskun Oregi, Iztok Fister, Iztok Fister, Javier Del Ser, "Hybridizing differential evolution and novelty search for multimodal optimization problems", Genetic and Evolutionary Computation Conference Companion, pp. 1980–1989, 2019.

6.3. Future Research Lines

Several research lines can be outlined departing from the insights, lessons learned and identified niches during the course of investigation:

- To begin with, the Thesis has identified several grand challenges in the wide fields of EM and EDL. Evolutionary trainable parameter optimization (model training) is among the problems in DL that requires rethinking what has to be optimized, and how. Interestingly, evolutionary model training has given rise to trained models that quickly overfit the training data. A question emerges at this point: does this result mean that the evolutionary algorithm in use is a worse solver for the problem at hand, or instead a call for reflection around what has to be optimized? Or is it a worse performance of gradient backpropagation algorithms an implicit way of regularizing the trained model? This question lacks a clear answer. In this regard, methods that blend easily into the evolutionary search could effectively overcome this observed phenomenon by avoiding an overfitting regime of the evolved parameters during the evolutionary process. The lottery ticket hypothesis [562], pruning methods over the search [563] or including learnable parametric formulations of the loss function to be optimized [564] can be interesting research directions to follow.

- Another caveat discussed in the Thesis is the computational cost associated to the use of evolutionary algorithms for trainable parameter optimization process, which can be quite demanding for large and complex neural networks. To address these limitations, the hybridization of evolutionary computation with local search heuristics that exploit the layered structure of neural networks can be investigated. Experiments done in this Thesis around scheduling the evolutionary search across the layers of the DL model (Section 2.6.4 in Chapter 2) have not been conclusive. Nevertheless, more elaborated schedules can be proposed, asynchronously leaping between layers based on the stagnation of the evolutionary search for every layer. In what refers to the computational cost, we foresee that the increasing relevance of EDL will spur the creation of software libraries with implementations of evolutionary solvers suitable to be deployed in multi-core computing devices, including graphical (GPU) and tensor (TPU) processing units. This, together with a parallel deployment of the fitness function to be evaluated for every use case, will guarantee the efficiency required for EDL to scale up nicely and become a realistic choice.

- Although the Thesis has dealt with two different modeling tasks (supervised and RL), the applicability of evolutionary computation and EM to other problems can be also worthwhile to be explored, as it has been analyzed in the literature review carried out in Section 2.3 (Figure 2.3). When learning to synthesize new examples from data distributions that resemble each other, one could expect that early modeled knowledge should be transferred among different generative models. In an envisioned setup comprising simultaneously evolved generative neural neworks (GANs), knowledge that is common to the training dataset of each task should be exploitable through EM, reaching higher levels of quality in the synthetic samples in potentially shorter training latencies.

- When focusing on multitask and meta-RL, the behavioral diversity of evolved agents could be added as another objective to be optimized through multi-factorial techniques. In this vein, quality-diversity optimization [565] could span new possibilities to jointly consider the fitness of evolved agents (i.e., their proficiency when solving their RL task) and the generation of models that learn to solve them differently (diversity). This could help the evolutionary process arrive at diverse solutions that represent different ways in which each task can be solved, even by exploiting tweaks in the implementation of the environments as the ones detected in this Thesis.

- Despite its undoubted theoretical utility, the assumptions made in zero-shot and few-shot learning about the unseen tasks can be often relaxed in practice. It is often the case that some meta-knowledge about test tasks is available before the meta-learning model attempts to solve them. The form in which this meta-knowledge is realized is largely application-dependent. For instance, in industrial prognosis it is unrealistic to think that the prognostic model will be asked to predict the probability of failure of a new machine whose purpose is absolutely different than the other machines existing in the plant. However, one may know the specific purpose for which the new machine will be placed in production. This meta-knowledge can be used to tailor how the meta-learning model tackles it, especially in the zero-shot learning scenario. In regard to the AML-MFEA approach proposed in Chapter 5, we envision that this meta-knowledge could be used to refine the weights associated to the policy averaging mechanism of the meta-task ensemble.

- Finally, further investigations are needed to endow EM methods for multi-task and meta-RL with the functionalities expected for more demanding modeling scenarios, including continual learning (knowledge retention, persistence and retrieval over time), open-world recognition/learning (rejection, characterization and consolidation of unknown knowledge into the model) and GPAI (multi-modal data and heterogeneous learning tasks). In all these paradigms the evolvability of the model is a must to accommodate the changes in terms of the task and/or the samples/data that the machine collects from its environment. Furthermore, TL is a key for knowledge characterization, storage and retrieval, since the commonalities between prevalent tasks can surely bias what knowledge must be persisted and recovered over time. These functional requirements expected for the next generation of Artificial Intelligence models unleashes a promising playground for EM in forthcoming years.

BIBLIOGRAPHY

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep Learning.» In: *Nature* 521.7553 (2015), pp. 436–444.

[2] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. «Backpropagation applied to handwritten zip code recognition.» In: *Neural computation* 1.4 (1989), pp. 541–551.

[3] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. «A fast learning algorithm for deep belief nets.» In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[4] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. «Recent trends in deep learning based natural language processing.» In: *IEEE Computational Intelligence Magazine* 13.3 (2018), pp. 55–75.

[5] Morten Kolbk, Zheng-Hua Tan, Jesper Jensen, Morten Kolbk, Zheng-Hua Tan, and Jesper Jensen. «Speech intelligibility potential of general and specialized deep neural network based speech enhancement systems.» In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* 25.1 (2017), pp. 153–167.

[6] Yu Zhang, William Chan, and Navdeep Jaitly. «Very deep convolutional networks for end-to-end speech recognition.» In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2017, pp. 4845–4849.

[7] Siddharth Pal, Yuxiao Dong, Bishal Thapa, Nitesh V Chawla, Ananthram Swami, and Ram Ramanathan. «Deep learning for network analysis: Problems, approaches and challenges.» In: *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE. 2016, pp. 588–593.

[8] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. «A survey of deep learning techniques for autonomous driving.» In: *Journal of Field Robotics* (2019).

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks.» In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. «Learning phrase representations using RNN encoder-decoder for statistical machine translation.» In: *arXiv preprint arXiv:1406.1078* (2014).

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. «Generative adversarial nets.» In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[12] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. «Deep learning applications and challenges in big data analytics.» In: *Journal of Big Data* 2.1 (2015), p. 1.

[13] Kai Yu, Lei Jia, Yuqiang Chen, and W Xu. «Deep learning: yesterday, today, and tomorrow.» In: *Journal of Computer Research and Development* 50.9 (2013), pp. 1799–1804.

[14] Simon Fong, Suash Deb, and Xin-she Yang. «How meta-heuristic algorithms contribute to deep learning in the hype of big data analytics.» In: *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*. 2018, pp. 3–25.

[15] Alejandro Barredo Arrieta et al. «Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI.» In: *Information Fusion* 58 (2020), pp. 82–115.

[16] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. «A survey of transfer learning.» In: *Journal of Big data* 3.1 (2016), p. 9.

[17] Yew-Soon Ong and Abhishek Gupta. «Evolutionary multitasking: a computer science view of cognitive multitasking.» In: *Cognitive Computation* 8.2 (2016), pp. 125–142.

[18] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. «Meta-learning in neural networks: A survey.» In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.9 (2021), pp. 5149–5169.

[19] Javier Del Ser, Eneko Osaba, Daniel Molina, Xin-She Yang, Sancho Salcedo-Sanz, David Camacho, Swagatam Das, Ponnuthurai N Suganthan, Carlos A Coello Coello, and Francisco Herrera. «Bio-inspired computation: Where we stand and what's next.» In: *Swarm and Evolutionary Computation* 48 (2019), pp. 220–250.

[20] Daniel Molina, Javier Poyatos, Javier Del Ser, Salvador García, Amir Hussain, and Francisco Herrera. «Comprehensive Taxonomies of Nature-and Bio-inspired Optimization: Inspiration versus Algorithmic Behavior, Critical Analysis and Recommendations.» In: *arXiv preprint arXiv:2002.08136* (2020).

[21] Ilhem BoussaïD, Julien Lepagnot, and Patrick Siarry. «A survey on optimization metaheuristics.» In: *Information Sciences* 237 (2013), pp. 82–117.

[22] Fred Glover and Manuel Laguna. «Tabu search.» In: *Handbook of combinatorial optimization*. 1998, pp. 2093–2229.

[23] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. «Optimization by simulated annealing.» In: *Science* 220.4598 (1983), pp. 671–680.

[24] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. 1989.

[25] K.A De Jong. «Analysis of the behavior of a class of genetic adaptive systems.» PhD thesis. University of Michigan, Michigan, USA, 1975.

[26] Marco Dorigo and Mauro Birattari. *Ant colony optimization*. 2010.

[27] James Kennedy, Russell Eberhart, et al. «Particle swarm optimization.» In: *IEEE international conference on neural networks*. Vol. 4. Perth, Australia. 1995, pp. 1942–1948.

[28] Esmaeil Atashpaz-Gargari and Caro Lucas. «Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition.» In: *2007 IEEE congress on evolutionary computation*. IEEE. 2007, pp. 4661–4667.

[29] Dervis Karaboga and Bahriye Basturk. «A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm.» In: *Journal of Global Optimization* 39.3 (2007), pp. 459–471.

[30] Xin-She Yang, Zhihua Cui, Renbin Xiao, Amir Hossein Gandomi, and Mehmet Karamanoglu. *Swarm intelligence and bio-inspired computation: theory and applications*. 2013.

[31] Pablo Moscato et al. «On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms.» In: *Caltech Concurrent Computation Program* 826 (1989), p. 1989.

[32] Ferrante Neri and Carlos Cotta. «Memetic algorithms and memetic computing optimization: A literature review.» In: *Swarm and Evolutionary Computation* 2 (2012), pp. 1–14.

[33] Daniel Molina, Antonio LaTorre, and Francisco Herrera. «SHADE with iterative local search for large-scale global optimization.» In: *2018 IEEE Congress on Evolutionary Computation*. IEEE. 2018, pp. 1–8.

[34] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. «Multifactorial evolution: toward evolutionary multitasking.» In: *IEEE Transactions on Evolutionary Computation* 20.3 (2015), pp. 343–357.

[35] Xin Yao and Yong Liu. «A new evolutionary system for evolving artificial neural networks.» In: *IEEE Transactions on Neural Networks* 8.3 (1997), pp. 694–713.

[36] Kenneth O Stanley and Risto Miikkulainen. «Evolving neural networks through augmenting topologies.» In: *Evolutionary computation* 10.2 (2002), pp. 99–127.

[37] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. «A hypercube-based encoding for evolving large-scale neural networks.» In: *Artificial life* 15.2 (2009), pp. 185–212.

[38] Sebastian Risi, Joel Lehman, and Kenneth O Stanley. «Evolving the placement and density of neurons in the hyperneat substrate.» In: *Conference on Genetic and evolutionary computation*. 2010, pp. 563–570.

[39] Risto Miikkulainen et al. «Evolving Deep Neural Networks.» In: *ArXiv* abs/1703.00548 (2017).

[40] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. «Large-scale evolution of image classifiers.» In: *International Conference on Machine Learning*. 2017, pp. 2902–2911.

[41] Thomas Swearingen, Will Drevo, Bennett Cyphers, Alfredo Cuesta-Infante, Arun Ross, and Kalyan Veeramachaneni. «ATM: A distributed, collaborative, scalable system for automated machine learning.» In: *2017 IEEE International Conference on Big Data*. IEEE. 2017, pp. 151–162.

[42] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. «Designing Neural Network Architectures using Reinforcement Learning.» In: *International Conference on Learning Representations* (2017).

[43] Joe Davison. *DEvol-Deep Neural Network Evolution*. 2017.

[44] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. «A genetic programming approach to designing convolutional neural network architectures.» In: *Genetic and Evolutionary Computation Conference*. ACM. 2017, pp. 497–504.

[45] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. «Adanet: Adaptive structural learning of artificial neural networks.» In: *International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 874–883.

[46] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. «Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning.» In: *arXiv preprint arXiv:1712.06567* (2017).

[47] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. «Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents.» In: *Advances in neural information processing systems*. 2018, pp. 5027–5038.

[48] Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, Matthias Urban, Michael Burkart, Max Dippel, Marius Lindauer, and Frank Hutter. «Towards Automatically-Tuned Deep Neural Networks.» In: *AutoML: Methods, Sytems, Challenges*. Ed. by Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. To appear. Dec. 2018. Chap. 7, pp. 141–156.

[49] Julián Muñoz-Ordóñez, Carlos Cobos, Martha Mendoza, Enrique Herrera-Viedma, Francisco Herrera, and Siham Tabik. «Framework for the training of deep neural networks in tensorflow using metaheuristics.» In: *International Conference on Intelligent Data Engineering and Automated Learning*. Springer. 2018, pp. 801–811.

[50] Alejandro Martín, Raúl Lara-Cabrera, Félix Fuentes-Hurtado, Valery Naranjo, and David Camacho. «EvoDeep: a new evolutionary approach for automatic deep neural networks parametrisation.» In: *Journal of Parallel and Distributed Computing* 117 (2018), pp. 180–191.

[51] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. «Efficient neural architecture search via parameter sharing.» In: *arXiv preprint arXiv:1802.03268* (2018).

[52] Jiayi Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. «Auptimizer-an Extensible, Open-Source Framework for Hyperparameter Tuning.» In: *2019 IEEE International Conference on Big Data*. IEEE. 2019, pp. 339–348.

[53] Filipe Assunçao, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. «DENSER: deep evolutionary network structured representation.» In: *Genetic Programming and Evolvable Machines* 20.1 (2019), pp. 5–35.

[54] Google. «Google Cloud AutoML.» In: (). URL: `https://cloud.google.com/automl/`.

[55] Haifeng Jin, Qingquan Song, and Xia Hu. «Auto-keras: An efficient neural architecture search system.» In: *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 1946–1956.

[56] Jason Liang, Elliot Meyerson, Babak Hodjat, Dan Fink, Karl Mutch, and Risto Miikkulainen. «Evolutionary neural automl for deep learning.» In: *Genetic and Evolutionary Computation Conference*. 2019, pp. 401–409.

[57] Piero Molino, Yaroslav Dudin, and Sai Sumanth Miryala. *Ludwig: a type-based declarative deep learning toolbox*. 2019. eprint: arXiv:1909.07930.

[58] Jonas da Silveira Bohrer, Bruno Iochins Grisci, and Marcio Dorn. *Neuroevolution of Neural Network Architectures Using CoDeepNEAT and Keras*. 2020. arXiv: 2002.04634 [cs.NE].

[59] Francisco Charte, Antonio J Rivera, Francisco Martínez, and María J del Jesus. «EvoAAA: An evolutionary methodology for automated neural autoencoder architecture search.» In: *Integrated Computer-Aided Engineering* Preprint (2020), pp. 1–21.

[60] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. «Evolving competitive car controllers for racing games with neuroevolution.» In: *Conference on Genetic and evolutionary computation*. ACM. 2009, pp. 1179–1186.

[61] Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. «Real-time neuroevolution in the NERO video game.» In: *IEEE Transactions on Evolutionary Computation* 9.6 (2005), pp. 653–668.

[62] Phillip Verbancsics and Josh Harguess. «Generative neuroevolution for deep learning.» In: *arXiv preprint arXiv:1312.5355* (2013).

[63] Xiao-Zhi Gao, Jing Wang, Jarno MA Tanskanen, Rongfang Bie, and Ping Guo. «BP neural networks with harmony search method-based training for epileptic EEG signal classification.» In: *2012 Eighth International Conference on Computational Intelligence and Security*. IEEE. 2012, pp. 252–257.

[64] Juan Peralta Donate, Xiaodong Li, Germán Gutiérrez Sánchez, and Araceli Sanchis de Miguel. «Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm.» In: *Neural Computing and Applications* 22.1 (2013), pp. 11–20.

[65] Xiaoyu Mao, Adriaan Ter Mors, Nico Roos, and Cees Witteveen. «Using neuro-evolution in aircraft deicing scheduling.» In: *Adaptive and Learning Agents and Multi-Agent Systems* (2007), pp. 138–145.

[66] Gregory Morse and Kenneth O Stanley. «Simple evolutionary optimization can rival stochastic gradient descent in neural networks.» In: *Genetic and Evolutionary Computation Conference 2016*. ACM. 2016, pp. 477–484.

[67] Karl Mason, Jim Duggan, and Enda Howley. «Neural network topology and weight optimization through neuro differential evolution.» In: *Genetic and Evolutionary Computation Conference Companion*. ACM. 2017, pp. 213–214.

[68] Haifeng Jin, Qingquan Song, and Xia Hu. «Efficient neural architecture search with network morphism.» In: *arXiv preprint arXiv:1806.10282* (2018).

[69] Varun Kumar Ojha, Ajith Abraham, and Václav Snášel. «Metaheuristic design of feedforward neural networks: A review of two decades of research.» In: *Engineering Applications of Artificial Intelligence* 60 (2017), pp. 97–116.

[70] Alejandro Baldominos, Yago Saez, and Pedro Isasi. «On the automated, evolutionary design of neural networks: past, present, and future.» In: *Neural Computing and Applications* (2019), pp. 1–27.

[71] Harith Al-Sahaf, Ying Bi, Qi Chen, Andrew Lensen, Yi Mei, Yanan Sun, Binh Tran, Bing Xue, and Mengjie Zhang. «A survey on evolutionary machine learning.» In: *Journal of the Royal Society of New Zealand* 49.2 (2019), pp. 205–228.

[72] Ashraf Darwish, Aboul Ella Hassanien, and Swagatam Das. «A survey of swarm and evolutionary computing approaches for deep learning.» In: *Artificial Intelligence Review* 53.3 (2020), pp. 1767–1812.

[73] Haruna Chiroma, Abdulsalam Ya'u Gital, Nadim Rana, M Abdulhamid Shafi'i, Amina N Muhammad, Aishatu Yahaya Umar, and Adamu I Abubakar. «Nature Inspired Meta-heuristic Algorithms for Deep Learning: Recent Progress and Novel Perspective.» In: *Science and Information Conference*. Springer. 2019, pp. 59–70.

[74] Xin He, Kaiyong Zhao, and Xiaowen Chu. «AutoML: A Survey of the State-of-the-Art.» In: *arXiv preprint arXiv:1908.00709* (2019).

[75] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, and Gary Yen. «A Survey on Evolutionary Neural Architecture Search.» In: *arXiv preprint arXiv:2008.10937* (2020).

[76] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. «Neural Architecture Search: A Survey.» In: *Journal of Machine Learning Research* 20 (2019), pp. 1–21.

[77] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. «Regularized evolution for image classifier architecture search.» In: *arXiv preprint arXiv:1802.01548* (2018).

[78] Ye-Hoon Kim, Bhargava Reddy, Sojung Yun, and Chanwon Seo. «Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy.» In: *JMLR: Workshop and Conference*. Vol. 1. 2017, pp. 1–8.

[79] Lingxi Xie and Alan Yuille. «Genetic cnn.» In: *IEEE International Conference on Computer Vision*. 2017, pp. 1379–1388.

[80] Zhichao Lu, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. «Multi-Criterion Evolutionary Design of Deep Convolutional Neural Networks.» In: *arXiv preprint arXiv:1912.01369* (2019).

[81] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. «NSGA-NET: a multi-objective genetic algorithm for neural architecture search.» In: *arXiv preprint arXiv:1810.03522* (2018).

[82] Pablo Ribalta Lorenzo and Jakub Nalepa. «Memetic evolution of deep neural networks.» In: *Genetic and Evolutionary Computation Conference*. 2018, pp. 505–512.

[83] Zefeng Chen, Yuren Zhou, and Zhengxin Huang. «Auto-creation of Effective Neural Network Architecture by Evolutionary Algorithm and ResNet for Image Classification.» In: *2019 IEEE International Conference on Systems, Man and Cybernetics*. IEEE. 2019, pp. 3895–3900.

[84] Benjamin Evans, Harith Al-Sahaf, Bing Xue, and Mengjie Zhang. «Evolutionary deep learning: A genetic programming approach to image classification.» In: *2018 IEEE Congress on Evolutionary Computation*. IEEE. 2018, pp. 1–6.

[85] Mohammad Javad Shafiee, Akshaya Mishra, and Alexander Wong. «Deep learning with Darwin: Evolutionary synthesis of deep neural networks.» In: *Neural Processing Letters* 48.1 (2018), pp. 603–613.

[86] Hangyu Zhu and Yaochu Jin. «Real-time Federated Evolutionary Neural Architecture Search.» In: *arXiv preprint arXiv:2003.02793* (2020).

[87] Travis Desell. «Large scale evolution of convolutional neural networks using volunteer computing.» In: *Genetic and Evolutionary Computation Conference Companion*. 2017, pp. 127–128.

[88] Hojjat Salehinejad and Shahrokh Valaee. «EDropout: Energy-Based Dropout and Pruning of Deep Neural Networks.» In: *arXiv preprint arXiv:2006.04270* (2020).

[89] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. «Hierarchical representations for efficient architecture search.» In: *arXiv preprint arXiv:1711.00436* (2017).

[90] Kefan Chen and Wei Pang. «ImmuNetNAS: An Immune-network approach for searching Convolutional Neural Network Architectures.» In: *arXiv preprint arXiv:2002.12704* (2020).

[91] Haoyu Zhang, Yaochu Jin, Ran Cheng, and Kuangrong Hao. «Sampled Training and Node Inheritance for Fast Evolutionary Neural Architecture Search.» In: *arXiv preprint arXiv:2003.11613* (2020).

[92] Amr AbdelFatah Ahmed, Saad M Saad Darwish, and Mohamed M El-Sherbiny. «A novel automatic CNN architecture design approach based on genetic algorithm.» In: *International Conference on Advanced Intelligent Systems and Informatics*. Springer. 2019, pp. 473–482.

[93] Bin Wang, Yanan Sun, Bing Xue, and Mengjie Zhang. «Evolving Deep Neural Networks by Multi-objective Particle Swarm Optimization for Image Classification.» In: *Genetic and Evolutionary Computation Conference* (2019).

[94] Bin Wang, Bing Xue, and Mengjie Zhang. «Particle Swarm Optimisation for Evolving Deep Neural Networks for Image Classification by Evolving and Stacking Transferable Blocks.» In: *arXiv preprint arXiv:1907.12659* (2019).

[95] Bin Wang, Yanan Sun, Bing Xue, and Mengjie Zhang. «A Hybrid GA-PSO Method for Evolving Architecture and Short Connections of Deep Convolutional Neural Networks.» In: *Pacific Rim International Conference on Artificial Intelligence*. Springer. 2019, pp. 650–663.

[96] Ya-Lan Hu and Liang Chen. «A nonlinear hybrid wind speed forecasting model using LSTM network, hysteretic ELM and Differential Evolution algorithm.» In: *Energy Conversion and Management* 173 (2018), pp. 123–142.

[97] Aditya Rawal and Risto Miikkulainen. «From nodes to networks: Evolving recurrent neural networks.» In: *arXiv preprint arXiv:1803.04439* (2018).

[98] P. J. Angeline, G. M. Saunders, and J. B. Pollack. «An evolutionary algorithm that constructs recurrent neural networks.» In: *IEEE Transactions on Neural Networks* 5.1 (1994), pp. 54–65.

[99] Amir Behjat, Sharat Chidambaran, and Souma Chowdhury. «Adaptive genomic evolution of neural network topologies (agent) for state-to-action mapping in autonomous agents.» In: *2019 International Conference on Robotics and Automation*. IEEE. 2019, pp. 9638–9644.

[100] Alexander Ororbia, AbdElRahman ElSaid, and Travis Desell. «Investigating recurrent neural network memory structures using neuro-evolution.» In: *Genetic and Evolutionary Computation Conference*. 2019, pp. 446–455.

[101] AbdElRahman ElSaid, Steven Benson, Shuchita Patwardhan, David Stadem, and Travis Desell. «Evolving recurrent neural networks for time series data prediction of coal plant parameters.» In: *International Conference on the Applications of Evolutionary Computation*. Springer. 2019, pp. 488–503.

[102] Andrés Camero, Jamal Toutouh, and Enrique Alba. «A specialized evolutionary strategy using mean absolute error random sampling to design recurrent neural networks.» In: *arXiv preprint arXiv:1909.02425* (2019).

[103] Travis Desell, Sophine Clachar, James Higgins, and Brandon Wild. «Evolving deep recurrent neural networks using ant colony optimization.» In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer. 2015, pp. 86–98.

[104] AbdElRahman A ElSaid, Alexander G Ororbia, and Travis J Desell. «The ant swarm neuro-evolution procedure for optimizing recurrent networks.» In: *arXiv preprint arXiv:1909.11849* (2019).

[105] AbdElRahman ElSaid, Alexander G Ororbia, and Travis J Desell. «Ant-based Neural Topology Search (ANTS) for Optimizing Recurrent Networks.» In: *International Conference on the Applications of Evolutionary Computation*. Springer. 2020, pp. 626–641.

[106] Chia-Feng Juang. «A hybrid of genetic algorithm and particle swarm optimization for recurrent network design.» In: *IEEE Transactions on Systems, Man, and Cybernetics* 34.2 (2004), pp. 997–1006.

[107] Filipe Assuncao, David Sereno, Nuno Lourenco, Penousal Machado, and Bernardete Ribeiro. «Automatic Evolution of AutoEncoders for Compressed Representations.» In: *2018 IEEE Congress on Evolutionary Computation*. IEEE. 2018, pp. 1–8.

[108] Sean Lander and Yi Shang. «EvoAE–A new evolutionary method for training autoencoders for deep learning networks.» In: *2015 IEEE 39th Annual Computer Software and Applications Conference*. Vol. 2. IEEE. 2015, pp. 790–795.

[109] Zhun Fan, Jiahong Wei, Guijie Zhu, Jiajie Mo, and Wenji Li. «Evolutionary Neural Architecture Search for Retinal Vessel Segmentation.» In: *arXiv* (2020), arXiv–2001.

[110] Lino Rodriguez-Coayahuitl, Alicia Morales-Reyes, and Hugo Jair Escalante. «Evolving autoencoding structures through genetic programming.» In: *Genetic Programming and Evolvable Machines* 20.3 (2019), pp. 413–440.

[111] Kai Liu, Li Min Zhang, and Yong Wei Sun. «Deep Boltzmann machines aided design based on genetic algorithms.» In: *Applied Mechanics and Materials*. Vol. 568. Trans Tech Publ. 2014, pp. 848–851.

[112] Jae Kwon Kim, Young Shin Han, and Jong Sik Lee. «Particle swarm optimization–deep belief network–based rare class prediction model for highly class imbalance problem.» In: *Concurrency and Computation: Practice and Experience* 29.11 (2017), e4128.

[113] Kaitav Nayankumar Mehta. «Neuroevolutionary Training of Deep Convolutional Generative Adversarial Networks.» In: (2019).

[114] Victor Costa, Nuno Lourenço, and Penousal Machado. «Coevolution of Generative Adversarial Networks.» In: *International Conference on the Applications of Evolutionary Computation*. Springer. 2019, pp. 473–487.

[115] Victor Costa, Nuno Lourenço, João Correia, and Penousal Machado. «COEGAN: evaluating the coevolution effect in generative adversarial networks.» In: *Genetic and Evolutionary Computation Conference*. 2019, pp. 374–382.

[116] Andreas Precht Poulsen, Mark Thorhauge, Mikkel Hvilshj Funch, and Sebastian Risi. «DLNE: A hybridization of deep learning and neuroevolution for visual control.» In: *2017 IEEE Conference on Computational Intelligence and Games*. IEEE. 2017, pp. 256–263.

[117] Son Pham, Keyi Zhang, Tung Phan, Jasper Ding, and Christopher L Dancy. «Playing SNES Games with NeuroEvolution of Augmenting Topologies.» In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[118] Kenneth O Stanley and Risto Miikkulainen. «Efficient reinforcement learning through evolving neural network topologies.» In: *Conference on Genetic and Evolutionary Computation*. 2002, pp. 569–577.

[119] Matthew Hausknecht, Joel Lehman, Risto Miikkulainen, and Peter Stone. «A neuroevolution approach to general atari game playing.» In: *IEEE Transactions on Computational Intelligence and AI in Games* 6.4 (2014), pp. 355–366.

[120] Jörg KH Franke, Gregor Köhler, Noor Awad, and Frank Hutter. «Neural Architecture Evolution in Deep Reinforcement Learning for Continuous Control.» In: *arXiv preprint arXiv:1910.12824* (2019).

[121] Etor Arza, Josu Ceberio, Aritz Pérez, and Ekhiñe Irurozki. «An adaptive neuroevolution-based hyperheuristic.» In: *Genetic and Evolutionary Computation Conference Companion*. 2020, pp. 111–112.

[122] Saya Fujino, Naoki Mori, and Keinosuke Matsumoto. «Deep convolutional networks for human sketches by means of the evolutionary deep learning.» In: *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems*. IEEE. 2017, pp. 1–5.

[123] Alejandro Baldominos, Yago Saez, and Pedro Isasi. «Evolutionary convolutional neural networks: An application to handwriting recognition.» In: *Neurocomputing* 283 (2018), pp. 38–52.

[124] Nasimul Noman Ali Bakhshi Stephan Chalup. «Fast Evolution of CNN Architecture for Image Classification.» In: *Deep Neural Evolution*. Ed. by Nasimul Noman Hitoshi Iba. 2020. Chap. 8, pp. 209–229.

[125] Rohan Akut and Siddhivinayak Kulkarni. «NeuroEvolution: Using Genetic Algorithm for optimal design of Deep Learning models.» In: *2019 IEEE International Conference on Electrical, Computer and Communication Technologies*. IEEE. 2019, pp. 1–6.

[126] Filipe Assunção, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. «Fast denser: Efficient deep neuroevolution.» In: *European Conference on Genetic Programming*. Springer. 2019, pp. 197–212.

[127] Erik Bochinski, Tobias Senst, and Thomas Sikora. «Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms.» In: *2017 IEEE International Conference on Image Processing*. IEEE. 2017, pp. 3924–3928.

[128] Jonas Prellberg and Oliver Kramer. «Lamarckian evolution of convolutional neural networks.» In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2018, pp. 424–435.

[129] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G Yen. «Automatically evolving cnn architectures based on blocks.» In: *arXiv preprint arXiv:1810.11875* (2018).

[130] Shuai Zhang, Yong Chen, XL Huang, and YS Cai. «Text Classification of Public Feedbacks using Convolutional Neural Network Based on Differential Evolution Algorithm.» In: *International Journal of Computers Communications & Control* 14.1 (2019), pp. 124–134.

[131] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. «Evolving deep neural networks.» In: *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. 2019, pp. 293–312.

[132] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. «Simple and efficient architecture search for convolutional neural networks.» In: *arXiv preprint arXiv:1711.04528* (2017).

[133] Benteng Ma, Xiang Li, Yong Xia, and Yanning Zhang. «Autonomous deep learning: A genetic DCNN designer for image classification.» In: *Neurocomputing* 379 (2020), pp. 152–161.

[134] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. «Designing Convolutional Neural Network Architectures Using Cartesian Genetic Programming.» In: *Deep Neural Evolution*. 2020, pp. 185–208.

[135] Xue Gu, Ziyao Meng, Yanchun Liang, Dong Xu, Han Huang, Xiaosong Han, and Chunguo Wu. «ESAE: Evolutionary Strategy-Based Architecture Evolution.» In: *International Conference on Bio-Inspired Computing: Theories and Applications*. Springer. 2019, pp. 193–208.

[136] Masanori Suganuma, Masayuki Kobayashi, Shinichi Shirakawa, and Tomoharu Nagao. «Evolution of deep convolutional neural networks using Cartesian genetic programming.» In: *Evolutionary Computation* 28.1 (2020), pp. 141–163.

[137] Hangyu Zhu and Yaochu Jin. «Multi-objective evolutionary federated learning.» In: *IEEE Transactions on Neural Networks and Learning Systems* (2019).

[138] Mohammad Loni, Sima Sinaei, Ali Zoljodi, Masoud Daneshtalab, and Mikael Sjödin. «DeepMaker: A Multi-Objective Optimization Framework for Deep Neural Networks in Embedded Systems.» In: *Microprocessors and Microsystems* (2020), p. 102989.

[139] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. «Efficient multi-objective neural architecture search via lamarckian evolution.» In: *arXiv preprint arXiv:1804.09081* (2018).

[140] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G Yen. «Evolving deep convolutional neural networks for image classification.» In: *IEEE Transactions on Evolutionary Computation* (2019).

[141] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G Yen. «Completely automated CNN architecture design based on blocks.» In: *IEEE Transactions on Neural Networks and Learning Systems* 31.4 (2019), pp. 1242–1254.

[142] Hojjat Rakhshani, Hassan Ismail, Lhassane Idoumghar, Germain Forestier, Julien Lepagnot, Jonathan Weber, Mathieu Brévilliers, and Pierre-Alain Muller. «Neural Architecture Search for Time Series Classification.» In: 2020 International Joint Conference on Neural Networks. 2020.

[143] Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. «NSGANetV2: Evolutionary Multi-Objective Surrogate-Assisted Neural Architecture Search.» In: *arXiv preprint arXiv:2007.10396* (2020).

[144] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. «Cars: Continuous evolution for efficient neural architecture search.» In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1829–1838.

[145] Yukang Chen, Gaofeng Meng, Qian Zhang, Shiming Xiang, Chang Huang, Lisen Mu, and Xinggang Wang. «Reinforced evolutionary neural architecture search.» In: *arXiv preprint arXiv:1808.00193* (2018).

[146] Hui Zhu, Zhulin An, Chuanguang Yang, Kaiqiang Xu, Erhu Zhao, and Yongjun Xu. «EENA: efficient evolution of neural architecture.» In: *IEEE International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[147] Yuqiao Liu, Yanan Sun, Bing Xue, and Mengjie Zhang. «Evolving Deep Convolutional Neural Networks for Hyperspectral Image Denoising.» In: *arXiv preprint arXiv:2008.06634* (2020).

[148] Maria G Baldeon Calisto and Susana K Lai-Yuen. «Self-adaptive 2D-3D ensemble of fully convolutional networks for medical image segmentation.» In: *Medical Imaging 2020: Image Processing*. Vol. 11313. International Society for Optics and Photonics. 2020, 113131W.

[149] Filipe Assunção, Nuno Lourenço, Bernardete Ribeiro, and Penousal Machado. «Incremental Evolution and Development of Deep Artificial Neural Networks.» In: *European Conference on Genetic Programming.* Springer. 2020, pp. 35–51.

[150] Binay Dahal and Justin Zhan. «Effective Mutation and Recombination for Evolving Convolutional Networks.» In: *Conference on Applications of Intelligent Systems*. 2020, pp. 1–6.

[151] Ahmed I Sharaf and El-Sayed F Radwan. «An Automated Approach for Developing a Convolutional Neural Network Using a Modified Firefly Algorithm for Image Classification.» In: *Applications of Firefly Algorithm and its Variants*. 2020, pp. 99–118.

[152] Dolly Sapra and Andy D Pimentel. «An evolutionary optimization algorithm for gradually saturating objective functions.» In: *Genetic and Evolutionary Computation Conference*. 2020.

[153] Dolly Sapra and Andy D Pimentel. «Constrained evolutionary piecemeal training to design convolutional neural networks.» In: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer*. 2020.

[154] Jing Jiang, Fei Han, Qinghua Ling, Jie Wang, Tiange Li, and Henry Han. «Efficient network architecture search via multiobjective particle swarm optimization based on decomposition.» In: *Neural Networks* 123 (2020), pp. 305–316.

[155] Fabio Marco Johner and Juergen Wassner. «Efficient Evolutionary Architecture Search for CNN Optimization on GTSRB.» In: *2019 18th IEEE International Conference On Machine Learning And Applications*. IEEE. 2019, pp. 56–61.

[156] Luc Frachon, Wei Pang, and George M Coghill. «ImmuNeCS: Neural Committee Search by an Artificial Immune System.» In: *arXiv preprint arXiv:1911.07729* (2019).

[157] Erfan Miahi, Seyed Abolghasem Mirroshandel, and Alexis Nasr. «Genetic Neural Architecture Search for automatic assessment of human sperm images.» In: *arXiv preprint arXiv:1909.09432* (2019).

[158] Danilo Vasconcellos Vargas and Shashank Kotyan. «Evolving Robust Neural Architectures to Defend from Adversarial Attacks.» In: *arXiv preprint arXiv:1906.11667* (2019).

[159] Shengyun Wei, Shun Zou, Feifan Liao, Weimin Lang, and Wenhui Wu. «Automatic Modulation Recognition Using Neural Architecture Search.» In: *2019 International Conference on High Performance Big Data and Intelligent Systems*. IEEE. 2019, pp. 151–156.

[160] Filipe Assunção, João Correia, Rúben Conceição, Mário João Martins Pimenta, Bernardo Tomé, Nuno Lourenço, and Penousal Machado. «Automatic design of artificial neural networks for Gamma-ray detection.» In: *IEEE Access* 7 (2019), pp. 110531–110540.

[161] Peng Liu, Mohammad D El Basha, Yangjunyi Li, Yao Xiao, Pina C Sanelli, and Ruogu Fang. «Deep evolutionary networks with expedited genetic algorithms for medical image denoising.» In: *Medical Image Analysis* 54 (2019), pp. 306–315.

[162] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, Jixiang Li, and Qingyuan Li. «Fast, accurate and lightweight super-resolution with neural architecture search.» In: *arXiv preprint arXiv:1901.07261* (2019).

[163] Chih-Che Chung, Wei-Ting Lin, Rong Zhang, Kai-Wen Liang, and Pao-Chi Chang. «Emotion Estimation by Joint Facial Expression and Speech Tonality Using Evolutionary Deep Learning Structures.» In: *2019 IEEE 8th Global Conference on Consumer Electronics*. IEEE. 2019, pp. 221–224.

[164] Ying Bi, Bing Xue, and Mengjie Zhang. «An Evolutionary Deep Learning Approach Using Genetic Programming with Convolution Operators for Image Classification.» In: *2019 IEEE Congress on Evolutionary Computation*. IEEE. 2019, pp. 3197–3204.

[165] Gerard Jacques van Wyk and Anna Sergeevna Bosman. «Evolutionary neural architecture search for image restoration.» In: *2019 International Joint Conference on Neural Networks*. IEEE. 2019, pp. 1–8.

[166] Elad Rapaport, Oren Shriki, and Rami Puzis. «EEGNAS: Neural Architecture Search for Electroencephalography Data Analysis and Decoding.» In: *International Workshop on Human Brain and Artificial Intelligence*. Springer. 2019, pp. 3–20.

[167] David Laredo, Yulin Qin, Oliver Schütze, and Jian-Qiao Sun. «Automatic Model Selection for Neural Networks.» In: *arXiv preprint arXiv:1905.06010* (2019).

[168] Edvinas Byla and Wei Pang. «Deepswarm: Optimising convolutional neural networks using swarm intelligence.» In: *UK Workshop on Computational Intelligence*. Springer. 2019, pp. 119–130.

[169] Jian Ren, Zhe Li, Jianchao Yang, Ning Xu, Tianbao Yang, and David J Foran. «Eigen: Ecologically-inspired genetic approach for neural network structure searching from scratch.» In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9059–9068.

[170] Dehua Song, Chang Xu, Xu Jia, Yiyi Chen, Chunjing Xu, and Yunhe Wang. «Efficient Residual Dense Block Search for Image Super-Resolution.» In.

[171] David Jones, Anja Schroeder, and Geoff Nitschke. «Evolutionary deep learning to identify galaxies in the zone of avoidance.» In: *arXiv preprint arXiv:1903.07461* (2019).

[172] Yukang Chen, Gaofeng Meng, Qian Zhang, Shiming Xiang, Chang Huang, Lisen Mu, and Xinggang Wang. «Renas: Reinforced evolutionary neural architecture search.» In: *IEEE Conference on computer vision and pattern recognition*. 2019, pp. 4787–4796.

[173] AJ Piergiovanni, Anelia Angelova, Alexander Toshev, and Michael S Ryoo. «Evolving space-time neural architectures for videos.» In: *IEEE international conference on computer vision*. 2019, pp. 1793–1802.

[174] Alejandro Martin, Raúl Lara-Cabrera, Víctor Manuel Vargas, Pedro Antonio Gutiérrez, César Hervás-Martínez, and David Camacho. «Statistically-driven Coral Reef metaheuristic for automatic hyperparameter setting and architecture design of Convolutional Neural Networks.» In: *2020 IEEE Congress on Evolutionary Computation*. IEEE. 2020, pp. 1–8.

[175] Emmanuel Dufourq and Bruce A Bassett. «Eden: Evolutionary deep networks for efficient machine learning.» In: *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics*. IEEE. 2017, pp. 110–115.

[176] Bin Wang, Yanan Sun, Bing Xue, and Mengjie Zhang. «Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification.» In: *2018 IEEE Congress on Evolutionary Computation*. IEEE. 2018, pp. 1–8.

[177] Mengjie Zhang Bin Wang Bing Xue. «Particle Swarm Optimization for Evolving Deep Convolutional Neural Networks for Image Classification: Single- and Multi-Objective Approaches.» In: *Deep Neural Evolution*. Ed. by Nasimul Noman Hitoshi Iba. 2020. Chap. 6, pp. 161–190.

[178] Tomaso Cetto, Jonathan Byrne, Xiaofan Xu, and David Moloney. «Size/Accuracy Trade-Off in Convolutional Neural Networks: An Evolutionary Approach.» In: *INNS Big Data and Deep Learning conference*. Springer. 2019, pp. 17–26.

[179] Vishal Passricha and Rajesh Kumar Aggarwal. «PSO-based optimized CNN for Hindi ASR.» In: *International Journal of Speech Technology* 22.4 (2019), pp. 1123–1133.

[180] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Hailong Ma. «Multiobjective reinforced evolution in mobile neural architecture search.» In: *arXiv preprint arXiv:1901.01074* (2019).

[181] Francisco Erivaldo Fernandes Junior and Gary G Yen. «Particle swarm optimization of deep neural networks architectures for image classification.» In: *Swarm and Evolutionary Computation* 49 (2019), pp. 62–74.

[182] Ben Fielding and Li Zhang. «Evolving image classification architectures with enhanced particle swarm optimisation.» In: *IEEE Access* 6 (2018), pp. 68560–68575.

[183] Bin Wang, Yanan Sun, Bing Xue, and Mengjie Zhang. «A hybrid differential evolution approach to designing deep convolutional neural networks for image classification.» In: *Australasian Joint Conference on Artificial Intelligence*. Springer. 2018, pp. 237–250.

[184] Lu Peng, Shan Liu, Rui Liu, and Lin Wang. «Effective long short-term memory with differential evolution algorithm for electricity price prediction.» In: *Energy* 162 (2018), pp. 1301–1314.

[185] Bahareh Nakisa, Mohammad Naim Rastgoo, Andry Rakotonirainy, Frederic Maire, and Vinod Chandran. «Long short term memory hyperparameter optimization for a neural network based emotion recognition framework.» In: *IEEE Access* 6 (2018), pp. 49325–49338.

[186] Aditya Rawal and Risto Miikkulainen. «Evolving deep LSTM-based memory networks using an information maximization objective.» In: *Genetic and Evolutionary Computation Conference*. 2016, pp. 501–508.

[187] Vicente Coelho Lobo Neto, Leandro Aparecido Passos, and João Paulo Papa. «Evolving long short-term memory networks.» In: *Conference on Computational Science–ICCS 2020*. Springer. 2020, pp. 337–350.

[188] Mehdi Neshat, Meysam Majidi Nezhad, Ehsan Abbasnejad, Lina Bertling Tjernberg, Davide Astiaso Garcia, Bradley Alexander, and Markus Wagner. «An evolutionary deep learning method for short-term wind speed prediction: A case study of the lillgrund offshore wind farm.» In: *arXiv preprint arXiv:2002.09106* (2020).

[189] Tomohiro Tanaka, Takafumi Moriya, Takahiro Shinozaki, Shinji Watanabe, Takaaki Hori, and Kevin Duh. «Automated structure discovery and parameter tuning of neural network language model based on evolution strategy.» In: *2016 IEEE Spoken Language Technology Workshop*. IEEE. 2016, pp. 665–671.

[190] Pedro Bento, José Pombo, Silvio Mariano, and Maria do Rosário Calado. «Short-Term Load Forecasting using optimized LSTM Networks via Improved Bat Algorithm.» In: *2018 International Conference on Intelligent Systems*. IEEE. 2018, pp. 351–357.

[191] Marijn van Knippenberg, Vlado Menkovski, and Sergio Consoli. «Evolutionary Construction of Convolutional Neural Networks.» In: *International Conference on Machine Learning, Optimization, and Data Science*. Springer. 2018, pp. 293–304.

[192] Francisco Charte, Antonio J Rivera, Francisco Martínez, and María J del Jesus. «Automating Autoencoder Architecture Configuration: An Evolutionary Approach.» In: *International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer. 2019, pp. 339–349.

[193] Kary Ho, Andrew Gilbert, Hailin Jin, and John Collomosse. «Neural Architecture Search for Deep Image Prior.» In: *arXiv preprint arXiv:2001.04776* (2020).

[194] Syahril Ramadhan Saufi, Zair Asrar bin Ahmad, Mohd Salman Leong, and Meng Hee Lim. «Differential evolution optimization for resilient stacked sparse autoencoder and its applications on bearing fault diagnosis.» In: *Measurement Science and Technology* 29.12 (2018), p. 125002.

[195] Masanori Suganuma, Mete Ozay, and Takayuki Okatani. «Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search.» In: *arXiv preprint arXiv:1803.00370* (2018).

[196] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G Yen. «A Particle swarm optimization-based flexible convolutional autoencoder for image classification.» In: *IEEE Transactions on Neural Networks and Learning Systems* (2018).

[197] Joao P Papa, Gustavo H Rosa, Aparecido N Marana, Walter Scheirer, and David D Cox. «Model selection for discriminative restricted boltzmann machines through meta-heuristic techniques.» In: *Journal of Computational Science* 9 (2015), pp. 14–18.

[198] Leandro Aparecido Passos and João Paulo Papa. «A metaheuristic-driven approach to fine-tune Deep Boltzmann Machines.» In: *Applied Soft Computing* (2019), p. 105717.

[199] Takashi Kuremoto, Shinsuke Kimura, Kunikazu Kobayashi, and Masanao Obayashi. «Time series forecasting using restricted boltzmann machine.» In: *International Conference on Intelligent Computing*. Springer. 2012, pp. 17–22.

[200] Leandro A Passos, Douglas R Rodrigues, and João P Papa. «Fine tuning deep boltzmann machines through meta-heuristic approaches.» In: *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics*. IEEE. 2018, pp. 000419–000424.

[201] Jinjiang Wang, Kebo Wang, Yangshen Wang, Zuguang Huang, and Ruijuan Xue. «Deep Boltzmann machine based condition prediction for smart manufacturing.» In: *Journal of Ambient Intelligence and Humanized Computing* 10.3 (2019), pp. 851–861.

[202] Nasser R Sabar, Ayad Turky, Andy Song, and Abdul Sattar. «Optimising Deep Belief Networks by hyper-heuristic approach.» In: *2017 IEEE Congress on Evolutionary Computation*. IEEE. 2017, pp. 2738–2745.

[203] Delowar Hossain, Genci Capi, and Mitsuru Jindai. «Evolution of Deep Belief Neural Network Parameters for Robot Object Recognition and Grasping.» In: *Procedia Computer Science* 105.C (2017), pp. 153–158.

[204] Gustavo H de Rosa and Joao P Papa. «Soft-Tempering Deep Belief Networks Parameters Through Genetic Programming.» In: (2019).

[205] Leandro Passos Júnior, Gustavo de Rosa, Douglas Rodrigues, Mateus Roder, and João Papa. «On the Assessment of Nature-Inspired Meta-Heuristic Optimization Techniques to Fine-Tune Deep Belief Networks.» In: May 2020, pp. 67–96. ISBN: 978-981-15-3684-7. DOI: 10.1007/978-981-15-3685-4_3.

[206] Nasser R Sabar, Ayad Turky, Andy Song, and Abdul Sattar. «An evolutionary hyper-heuristic to optimise deep belief networks for image reconstruction.» In: *Applied Soft Computing* (2019), p. 105510.

[207] Ming-Huwi Horng. «Fine-Tuning Parameters of Deep Belief Networks Using Artificial Bee Colony Algorithm.» In: *DEStech Transactions on Computer Science and Engineering* (2017).

[208] Linchao Li, Lingqiao Qin, Xu Qu, Jian Zhang, Yonggang Wang, and Bin Ran. «Day-ahead traffic flow forecasting based on a deep belief network optimized by the multi-objective particle swarm algorithm.» In: *Knowledge-Based Systems* (2019).

[209] Shidrokh Goudarzi, Mohd Kama, Mohammad Anisi, Seyed Soleymani, and Faiyaz Doctor. «Self-organizing traffic flow prediction with an optimized deep belief network for internet of vehicles.» In: *Sensors* 18.10 (2018), p. 3459.

[210] Meng Ma, Chuang Sun, and Xuefeng Chen. «Discriminative deep belief networks with ant colony optimization for health status assessment of machine.» In: *IEEE Transactions on Instrumentation and Measurement* 66.12 (2017), pp. 3115–3125.

[211] Takashi Kuremoto, Takaomi Hirata, Masanao Obayashi, Kunikazu Kobayashi, and Shingo Mabu. «Search Heuristics for the Optimization of DBN for Time Series Forecasting.» In: *Deep Neural Evolution*. 2020, pp. 131–152.

[212] D Rodrigues, X-S Yang, and JP Papa. «Fine-tuning deep belief networks using cuckoo search.» In: *Bio-Inspired Computation and Applications in Image Processing*. 2016, pp. 47–59.

[213] Unai Garciarena, Roberto Santana, and Alexander Mendiburu. «Evolved GANs for generating Pareto set approximations.» In: *Genetic and Evolutionary Computation Conference*. ACM. 2018, pp. 434–441.

[214] Yantao Lu, Burak Kakillioglu, and Senem Velipasalar. «Autonomously and Simultaneously Refining Deep Neural Network Parameters by a Bi-Generative Adversarial Network Aided Genetic Algorithm.» In: *arXiv preprint arXiv:1809.10244* (2018).

[215] Abdelghani Dahou, Mohamed Abd Elaziz, Junwei Zhou, and Shengwu Xiong. «Arabic Sentiment Classification Using Convolutional Neural Network and Differential Evolution Algorithm.» In: *Computational Intelligence and Neuroscience* 2019 (2019).

[216] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. «Optimizing deep learning hyper-parameters through an evolutionary algorithm.» In: *Workshop on Machine Learning in High-Performance Computing Environments*. ACM. 2015, p. 4.

[217] Garrett Bingham, William Macke, and Risto Miikkulainen. «Evolutionary optimization of deep learning activation functions.» In: *arXiv preprint arXiv:2002.07224* (2020).

[218] J. Kim and S. Cho. «Evolutionary Optimization of Hyperparameters in Deep Learning Models.» In: *2019 IEEE Congress on Evolutionary Computation*. 2019, pp. 831–837.

[219] Santiago Gonzalez and Risto Miikkulainen. *Improved Training Speed, Accuracy, and Data Utilization Through Loss Function Optimization*. 2020. arXiv: 1905.11528 [cs.LG].

[220] Han Shu and Yunhe Wang. «Automatically Searching for U-Net Image Translator Architecture.» In: *arXiv preprint arXiv:2002.11581* (2020).

[221] Animesh Singh, Sandip Saha, Ritesh Sarkhel, Mahantapas Kundu, Mita Nasipuri, and Nibaran Das. «A Genetic Algorithm based Kernel-size Selection Approach for a Multi-column Convolutional Neural Network.» In: *arXiv preprint arXiv:1912.12405* (2019).

[222] Pablo Ribalta Lorenzo, Jakub Nalepa, Michal Kawulok, Luciano Sanchez Ramos, and José Ranilla Pastor. «Particle swarm optimization for hyper-parameter selection in deep neural networks.» In: *Genetic and Evolutionary Computation Conference*. ACM. 2017, pp. 481–488.

[223] Toshihiko Yamasaki, Takuto Honma, and Kiyoharu Aizawa. «Efficient optimization of convolutional neural networks using particle swarm optimization.» In: *2017 IEEE Third International Conference on Multimedia Big Data*. IEEE. 2017, pp. 70–73.

[224] Patxi Ortego, Alberto Diez-Olivan, Javier Del Ser, Fernando Veiga, Mariluz Penalva, and Basilio Sierra. «Evolutionary LSTM-FCN networks for pattern classification in industrial processes.» In: *Swarm and Evolutionary Computation* 54 (2020), p. 100650.

[225] AbdElRahman ElSaid, Fatima El Jamiy, James Higgins, Brandon Wild, and Travis Desell. «Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration.» In: *Applied Soft Computing* 73 (2018), pp. 969–991.

[226] AbdElRahman ElSaid, Fatima El Jamiy, James Higgins, Brandon Wild, and Travis Desell. «Using ant colony optimization to optimize long short-term memory recurrent neural networks.» In: *Genetic and Evolutionary Computation Conference*. ACM. 2018, pp. 13–20.

[227] Heshan Wang and Xuefeng Yan. «Optimizing the echo state network with a binary particle swarm optimization algorithm.» In: *Knowledge-Based Systems* 86 (2015), pp. 182–193.

[228] Tim Silhan, Stefan Oehmcke, and Oliver Kramer. «Evolution of Stacked Autoencoders.» In: *2019 IEEE Congress on Evolutionary Computation*. IEEE. 2019, pp. 823–830.

[229] João Paulo Papa, Walter Scheirer, and David Daniel Cox. «Fine-tuning deep belief networks using harmony search.» In: *Applied Soft Computing* 46 (2016), pp. 875–885.

[230] João Paulo Papa, Gustavo H Rosa, Danillo R Pereira, and Xin-She Yang. «Quaternion-based deep belief networks fine-tuning.» In: *Applied Soft Computing* 60 (2017), pp. 328–335.

[231] Gustavo Rosa, João Papa, Kelton Costa, Leandro Passos, Clayton Pereira, and Xin-She Yang. «Learning parameters in deep belief networks through firefly algorithm.» In: *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer. 2016, pp. 138–149.

[232] Muneeb ul Hassan, Nasser R Sabar, and Andy Song. «Optimising Deep Learning by Hyper-heuristic Approach for Classifying Good Quality Images.» In: *International Conference on Computational Science*. Springer. 2018, pp. 528–539.

[233] Clayton R Pereira, Danillo R Pereira, Joao P Papa, Gustavo H Rosa, and Xin-She Yang. «Convolutional neural networks applied for parkinson's disease identification.» In: *Machine Learning for Health Informatics*. 2016, pp. 377–390.

[234] Gustavo H De Rosa, João P Papa, and Xin-S Yang. «Handling dropout probability estimation in convolution neural networks using meta-heuristics.» In: *Soft Computing* (2018), pp. 1–10.

[235] Teck Yan Tan, Li Zhang, Chee Peng Lim, Ben Fielding, Yonghong Yu, and Emma Anderson. «Evolving ensemble models for image segmentation using enhanced particle swarm optimization.» In: *IEEE Access* 7 (2019), pp. 34004–34019.

[236] Baosu Guo, Jingwen Hu, Wenwen Wu, Qingjin Peng, and Fenghe Wu. «The Tabu_Genetic Algorithm: A Novel Method for Hyper-Parameter Optimization of Learning Algorithms.» In: *Electronics* 8.5 (2019), p. 579.

[237] Amelia Ritahani Ismail, Omar Abdelaziz Mohammad, et al. «Evolutionary deep belief networks with bootstrap sampling for imbalanced class datasets.» In: *International Journal of Advances in Intelligent Informatics* 5.2 (2019), pp. 123–136.

[238] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. «Population based training of neural networks.» In: *arXiv preprint arXiv:1711.09846* (2017).

[239] Krzysztof Pawelczyk, Michal Kawulok, and Jakub Nalepa. «Genetically-trained Deep Neural Networks.» In: *Genetic and Evolutionary Computation Conference Companion*. 2018, pp. 63–64.

[240] LM Rere, Mohamad Ivan Fanany, and Aniati Murni Arymurthy. «Metaheuristic algorithms for convolution neural network.» In: *Computational Intelligence and Neuroscience* 2016 (2016).

[241] Lucian-Ovidiu Fedorovici, Radu-Emil Precup, Florin Dragan, Radu-Codrut David, and Constantin Purcaru. «Embedding gravitational search algorithms in convolutional neural networks for OCR applications.» In: *2012 7th IEEE International Symposium on Applied Computational Intelligence and Informatics*. IEEE. 2012, pp. 125–130.

[242] Alejandro Martín García, Víctor Vargas Yun, Pedro Antonio Gutiérrez, David Camacho, and Cesar Martínez. «Optimising Convolutional Neural Networks using a Hybrid Statistically-driven Coral Reef Optimisation algorithm.» In: *Applied Soft Computing* 90 (2020). DOI: 10.1016/j.asoc.2020.106144.

[243] Jiawei Zhang and Fisher B Gouza. «GADAM: genetic-evolutionary ADAM for deep neural network optimization.» In: *arXiv preprint arXiv:1805.07500* (2018).

[244] Xiaodong Cui, Wei Zhang, Zoltán Tüske, and Michael Picheny. «Evolutionary stochastic gradient descent for optimization of deep neural networks.» In: *Advances in neural information processing systems*. 2018, pp. 6048–6058.

[245] Vasco Lopes and Paulo Fazendeiro. «A Hybrid Method for Training Convolutional Neural Networks.» In: *arXiv preprint arXiv:2005.04153* (2020).

[246] Di Zang, Jianping Ding, Jiujun Cheng, Dongdong Zhang, and Keshuang Tang. «A Hybrid Learning Algorithm for the Optimization of Convolutional Neural Network.» In: *International Conference on Intelligent Computing*. Springer. 2017, pp. 694–705.

[247] Anan Banharnsakun. «Towards improving the convolutional neural networks for deep learning using the distributed artificial bee colony method.» In: *International Journal of Machine Learning and Cybernetics* (2018), pp. 1–11.

[248] Mujahid H Khalifa, Marwa Ammar, Wael Ouarda, and Adel M Alimi. «Particle swarm optimization for deep learning of convolution neural network.» In: *2017 Sudan Conference on Computer Science and Information Technology*. IEEE. 2017, pp. 1–5.

[249] Youru Li, Zhenfeng Zhu, Deqiang Kong, Hua Han, and Yao Zhao. «EA-LSTM: Evolutionary attention-based LSTM for time series prediction.» In: *Knowledge-Based Systems* 181 (2019), p. 104785.

[250] Nazri Mohd Nawi, Abdullah Khan, MZ Rehman, Haruna Chiroma, and Tutut Herawan. «Weight optimization in recurrent neural networks with hybrid metaheuristic Cuckoo search techniques for data classification.» In: *Mathematical Problems in Engineering* 2015 (2015).

[251] Samuel Alvernaz and Julian Togelius. «Autoencoder-augmented neuroevolution for visual doom playing.» In: *2017 IEEE Conference on Computational Intelligence and Games*. IEEE. 2017, pp. 1–8.

[252] Omid E David and Iddo Greental. «Genetic algorithms for evolving deep neural networks.» In: *Conference on Genetic and Evolutionary Computation*. 2014, pp. 1451–1452.

[253] Erez Levy, Omid E David, and Nathan S Netanyahu. «Genetic algorithms and deep learning for automatic painter classification.» In: *Conference on Genetic and Evolutionary Computation*. 2014, pp. 1143–1150.

[254] Chong Zhang, Pin Lim, A Kai Qin, and Kay Chen Tan. «Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics.» In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2016), pp. 2306–2318.

[255] Abdullah Al-Dujaili, Tom Schmiedlechner, Una-May O'Reilly, et al. «Towards distributed coevolutionary GANs.» In: *arXiv preprint arXiv:1807.08194* (2018).

[256] Shauharda Khadka and Kagan Tumer. «Evolutionary reinforcement learning.» In: *arXiv preprint arXiv:1805.07917* (2018).

[257] Shauharda Khadka, Somdeb Majumdar, Tarek Nassar, Zach Dwiel, Evren Tumer, Santiago Miret, Yinyin Liu, and Kagan Tumer. «Collaborative evolutionary reinforcement learning.» In: *arXiv preprint arXiv:1905.00976* (2019).

[258] Shauharda Khadka and Kagan Tumer. «Evolution-Guided Policy Gradient in Reinforcement Learning.» In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. 2018, pp. 1188–1200.

[259] Jan Koutník, Jürgen Schmidhuber, and Faustino Gomez. «Evolving deep unsupervised convolutional networks for vision-based reinforcement learning.» In: *Conference on Genetic and Evolutionary Computation*. ACM. 2014, pp. 541–548.

[260] LM Rasdi Rere, Mohamad Ivan Fanany, and Aniati Murni Arymurthy. «Simulated annealing algorithm for deep learning.» In: *Procedia Computer Science* 72 (2015), pp. 137–144.

[261] Cesar Affonso de Pinho Pinheiro, Nadia Nedjah, and Luiza de Macedo Mourelle. «Detection and classification of pulmonary nodules using deep learning and swarm intelligence.» In: *Multimedia Tools and Applications* (2019), pp. 1–29.

[262] Vina Ayumi, LM Rasdi Rere, Mohamad Ivan Fanany, and Aniati Murni Arymurthy. «Optimization of convolutional neural network using microcanonical annealing algorithm.» In: *Conference on Advanced Computer Science and Information Systems*. IEEE. 2016, pp. 506–511.

[263] Gustavo Rosa, Joao Papa, Aparecido Marana, Walter Scheirer, and David Cox. «Fine-tuning convolutional neural networks using harmony search.» In: *Iberoamerican Congress on Pattern Recognition*. Springer. 2015, pp. 683–690.

[264] Sebastian Risi and Kenneth O Stanley. «Deep Neuroevolution of Recurrent and Discrete World Models.» In: *arXiv preprint arXiv:1906.08857* (2019).

[265] Tarik A Rashid, Polla Fattah, and Delan K Awla. «Using Accuracy Measure for Improving the Training of LSTM with Metaheuristic Algorithms.» In: *Procedia Computer Science* 140 (2018), pp. 324–333.

[266] Tarik A Rashid, Mohammad K Hassan, Mokhtar Mohammadi, and Kym Fraser. «Improvement of Variant Adaptable LSTM Trained With Metaheuristic Algorithms for Healthcare Analysis.» In: *Advanced Classification Techniques for Healthcare Analysis*. 2019, pp. 111–131.

[267] Niels Van Hoorn, Julian Togelius, and Jurgen Schmidhuber. «Hierarchical controller learning in a first-person shooter.» In: *2009 IEEE symposium on computational intelligence and games*. IEEE. 2009, pp. 294–301.

[268] Carlos A Duchanoy, Marco A Moreno-Armendáriz, Leopoldo Urbina, Carlos A Cruz-Villar, Hiram Calvo, and J de J Rubio. «A novel recurrent neural network soft sensor via a differential evolution training algorithm for the tire contact patch.» In: *Neurocomputing* 235 (2017), pp. 71–82.

[269] Biswajit Jana, Suman Mitra, and Sriyankar Acharyaa. «Reconstruction of Gene Regulatory Network Using Recurrent Neural Network Model: A Harmony Search Approach.» In: *Soft Computing and Signal Processing*. 2019, pp. 129–138.

[270] Surama Biswas and Sriyankar Acharyya. «A Bi-objective RNN model to reconstruct gene regulatory network: a modified multi-objective simulated annealing approach.» In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 15.6 (2018), pp. 2053–2059.

[271] Amr M Ibrahim and Noha H El-Amary. «Particle Swarm Optimization trained recurrent neural network for voltage instability prediction.» In: *Journal of Electrical Systems and Information Technology* 5.2 (2018), pp. 216–228.

[272] H. Hisashi. «Deep Boltzmann Machine for evolutionary agents of Mario AI.» In: *2014 IEEE Congress on Evolutionary Computation.* 2014, pp. 36–41.

[273] Chia-Feng Juang, Yu-Cheng Chang, and I-Fang Chung. «Optimization of Recurrent Neural Networks Using Evolutionary Group-based Particle Swarm Optimization for Hexapod Robot Gait Generation.» In: *Hybrid Metaheuristics: Research And Applications* 84 (2018), p. 227.

[274] Qin Song, Yu-Jun Zheng, Yu Xue, Wei-Guo Sheng, and Mei-Rong Zhao. «An evolutionary deep neural network for predicting morbidity of gastrointestinal infections by food contamination.» In: *Neurocomputing* 226 (2017), pp. 16–22.

[275] Delowar Hossain and Genci Capi. «Multiobjective evolution of deep learning parameters for robot manipulator object recognition and grasping.» In: *Advanced Robotics* 32.20 (2018), pp. 1090–1101.

[276] Chaoyue Wang, Chang Xu, Xin Yao, and Dacheng Tao. «Evolutionary generative adversarial networks.» In: *IEEE Transactions on Evolutionary Computation* 23.6 (2019), pp. 921–934.

[277] Jamal Toutouh, Erik Hemberg, and Una-May O'Reilly. «Spatial evolutionary generative adversarial networks.» In: *Genetic and Evolutionary Computation Conference.* 2019, pp. 472–480.

[278] Juan Song, Yi Jin, YiDong Li, and Congyan Lang. «Learning Structural Similarity with Evolutionary-GAN: A New Face De-identification Method.» In: *2019 6th International Conference on Behavioral, Economic and Socio-Cultural Computing.* IEEE. 2019, pp. 1–6.

[279] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. «Accelerated neural evolution through cooperatively coevolved synapses.» In: *Journal of Machine Learning Research* 9.May (2008), pp. 937–965.

[280] Christian Igel. «Neuroevolution for reinforcement learning using evolution strategies.» In: *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.* Vol. 4. IEEE. 2003, pp. 2588–2595.

[281] Aritz D Martinez, Eneko Osaba, Javier Del Ser, and Francisco Herrera. «Simultaneously Evolving Deep Reinforcement Learning Models using Multifactorial Optimization.» In: *arXiv preprint arXiv:2002.12133* (2020).

[282] Karl Mason, Jim Duggan, and Enda Howley. «Maze navigation using neural networks evolved with novelty search and differential evolution.» In: *Adaptive and Learning Agents Workshop.* 2018.

[283] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. «Back to basics: Benchmarking canonical evolution strategies for playing atari.» In: *arXiv preprint arXiv:1802.08842* (2018).

[284] Siham Tabik, Ricardo F Alvear-Sandoval, María M Ruiz, José-Luis Sancho-Gómez, Aníbal R Figueiras-Vidal, and Francisco Herrera. «MNIST-NET10: A heterogeneous deep networks fusion based on the degree of certainty to reach 0.1% error rate. ensembles overview and proposal.» In: *Information Fusion* 62 (2020), pp. 1–8.

[285] Antonio LaTorre, Daniel Molina, Eneko Osaba, Javier Del Ser, and Francisco Herrera. «Fairness in Bio-inspired Optimization Research: A Prescription of Methodological Guidelines for Comparing Meta-heuristics.» In: *arXiv preprint arXiv:2004.09969* (2020).

[286] Janez Demšar. «Statistical comparisons of classifiers over multiple data sets.» In: *Journal of Machine Learning Research* 7.Jan (2006), pp. 1–30.

[287] J Carrasco, S García, MM Rueda, S Das, and F Herrera. «Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review.» In: *Swarm and Evolutionary Computation* 54 (2020), p. 100665.

[288] Laurence Moroney. *Horses or Humans Dataset*. 2019. URL: http://laurencemoroney.com/horses-or-humans-dataset.

[289] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. «Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.» In: *Clinical Orthopaedics and Related Research* abs/1703.10593 (2017). arXiv: 1703.10593. URL: http://arxiv.org/abs/1703.10593.

[290] Yann LeCun and Corinna Cortes. «MNIST handwritten digit database.» In: (2010).

[291] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. «CIFAR-10 (Canadian Institute for Advanced Research).» In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[292] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. «Nsga-net: neural architecture search using multi-objective genetic algorithm.» In: *Genetic and Evolutionary Computation Conference*. 2019, pp. 419–427.

[293] Tomás Mantecón, Carlos R del Blanco, Fernando Jaureguizar, and Narciso García. «Hand gesture recognition using infrared imagery provided by leap motion controller.» In: 2016.

[294] *Blood Cell Classification Dataset*. 2019. URL: https://github.com/Shenggan/BCCD_Dataset.

[295] Han Xiao, Kashif Rasul, and Roland Vollgraf. «Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.» In: *arXiv preprint arXiv:1708.07747* (2017).

[296] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. «Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition.» In: *Neural Networks* 0 (2012), pp. –.

[297] Hanxiao Liu, Karen Simonyan, and Yiming Yang. «DARTS: Differentiable Architecture Search.» In: *International Conference on Learning Representations*. 2018.

[298] Roxana Istrate, Florian Scheidegger, Giovanni Mariani, Dimitrios Nikolopoulos, Constantine Bekas, and Adelmo Cristiano Innocenza Malossi. «Tapas: Train-less accuracy predictor for architecture search.» In: *AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 3927–3934.

[299] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. «Accelerating neural architecture search using performance prediction.» In: *arXiv preprint arXiv:1705.10823* (2017).

[300] Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G Yen, and Mengjie Zhang. «Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor.» In: *IEEE Transactions on Evolutionary Computation* 24.2 (2019), pp. 350–364.

[301] Yanan Sun, Xian Sun, Yuhan Fang, and Gary Yen. *A Novel Training Protocol for Performance Predictors of Evolutionary Neural Architecture Search Algorithms*. 2020. eprint: arXiv:2008.13187.

[302] Tom Veniat and Ludovic Denoyer. «Learning time/memory-efficient deep architectures with budgeted super networks.» In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3492–3500.

[303] Mokhtar Essaid, Lhassane Idoumghar, Julien Lepagnot, and Mathieu Brévilliers. «GPU parallelization strategies for metaheuristics: a survey.» In: *International Journal of Parallel, Emergent and Distributed Systems* 34.5 (2019), pp. 497–522.

[304] Ying Tan and Ke Ding. «A survey on GPU-based implementation of swarm intelligence algorithms.» In: *IEEE Transactions on Cybernetics* 46.9 (2015), pp. 2028–2041.

[305] Guido Schryen. «Parallel computational optimization in operations research: A new integrative framework, literature review and research directions.» In: *European Journal of Operational Research* (2019).

[306] Antonio Benitez-Hidalgo, Antonio J Nebro, Jose Garcia-Nieto, Izaskun Oregi, and Javier Del Ser. «jMetalPy: A Python framework for multi-objective optimization with metaheuristics.» In: *Swarm and Evolutionary Computation* 51 (2019), p. 100598.

[307] Youssef SG Nashed, Roberto Ugolotti, Pablo Mesejo, and Stefano Cagnoni. «libCudaOptimize: an open source library of GPU-based metaheuristics.» In: *Conference Companion on Genetic and Evolutionary Computation*. 2012, pp. 117–124.

[308] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. «Tensorflow: a system for large-scale machine learning.» In.

[309] Prasanna Balaprakash, Michael Salim, Thomas Uram, Venkat Vishwanath, and Stefan Wild. «DeepHyper: Asynchronous hyperparameter search for deep neural networks.» In: *Conference on High Performance Computing*. IEEE. 2018, pp. 42–51.

[310] Sedigheh Mahdavi, Mohammad Ebrahim Shiri, and Shahryar Rahnamayan. «Metaheuristics in large-scale global continues optimization: A survey.» In: *Information Sciences* 295 (2015), pp. 407–428.

[311] Jiao-Hong Yi, Li-Ning Xing, Gai-Ge Wang, Junyu Dong, Athanasios V Vasilakos, Amir H Alavi, and Ling Wang. «Behavior of crossover operators in NSGA-III for large-scale optimization problems.» In: *Information Sciences* 509 (2020), pp. 470–487.

[312] Jason Liang, Elliot Meyerson, and Risto Miikkulainen. «Evolutionary architecture search for deep multitask networks.» In: *Genetic and Evolutionary Computation Conference*. 2018, pp. 466–473.

[313] Yangyang Li, Shuangkang Fang, Xiaoyu Bai, Licheng Jiao, and Naresh Marturi. «Parallel Design of Sparse Deep Belief Network with Multi-objective Optimization.» In: *Information Sciences* (2020).

[314] Rohitash Chandra, Abhishek Gupta, Yew-Soon Ong, and Chi-Keong Goh. «Evolutionary multi-task learning for modular knowledge representation in neural networks.» In: *Neural Processing Letters* 47.3 (2018), pp. 993–1009.

[315] Abhishek Gupta, Yew-Soon Ong, Liang Feng, and Kay Chen Tan. «Multiobjective multifactorial optimization in evolutionary multitasking.» In: *IEEE Transactions on Cybernetics* 47.7 (2016), pp. 1652–1665.

[316] Shuangshuang Yao, Zhiming Dong, Xianpeng Wang, and Lei Ren. «A Multiobjective multifactorial optimization algorithm based on decomposition and dynamic resource allocation strategy.» In: *Information Sciences* 511 (2020), pp. 18–35.

[317] Kavitesh Kumar Bali, Abhishek Gupta, Yew-Soon Ong, and Puay Siew Tan. «Cognizant Multitasking in Multiobjective Multifactorial Evolution: MO-MFEA-II.» In: *IEEE Transactions on Cybernetics* (2020).

[318] Mantas Lukoševičius and Herbert Jaeger. «Reservoir computing approaches to recurrent neural network training.» In: *Computer Science Review* 3.3 (2009), pp. 127–149.

[319] Matthew Dale. «Neuroevolution of hierarchical reservoir computers.» In: *Genetic and Evolutionary Computation Conference*. 2018, pp. 410–417.

[320] Yan Zhou, Yaochu Jin, and Jinliang Ding. «Evolutionary Optimization of Liquid State Machines for Robust Learning.» In: *International Symposium on Neural Networks*. Springer. 2019, pp. 389–398.

[321] Yan Zhou, Yaochu Jin, and Jinliang Ding. «Surrogate-Assisted Evolutionary Search of Spiking Neural Architectures in Liquid State Machines.» In: *Neurocomputing* (2020).

[322] Kai Liu and Jie Zhang. «Nonlinear Process Modelling Using Echo State Networks Optimised by Covariance Matrix Adaption Evolutionary Strategy.» In: *Computers & Chemical Engineering* (2020), p. 106730.

[323] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. «Deep reservoir computing: A critical experimental analysis.» In: *Neurocomputing* 268 (2017), pp. 87–99.

[324] Roberto A Vazquez. «Training spiking neural models using cuckoo search algorithm.» In: *2011 IEEE Congress of Evolutionary Computation*. IEEE. 2011, pp. 679–686.

[325] Catherine D Schuman, James S Plank, Adam Disney, and John Reynolds. «An evolutionary optimization framework for neural networks and neuromorphic architectures.» In: *2016 International Joint Conference on Neural Networks*. IEEE. 2016, pp. 145–154.

[326] Roberto A Vazquez and Beatriz A Garro. «Training spiking neural models using artificial bee colony.» In: *Computational Intelligence and Neuroscience* 2015 (2015).

[327] Ruben Carino-Escobar, Jessica Cantillo-Negrete, Roberto A Vazquez, and Josefina Gutierrez-Martinez. «Spiking neural networks trained with particle swarm optimization for motor imagery classification.» In: *International Conference on Swarm Intelligence*. Springer. 2016, pp. 245–252.

[328] Xiangwen Wang, Xianghong Lin, and Xiaochao Dang. «Supervised learning in spiking neural networks: A review of algorithms and evaluations.» In: *Neural Networks* (2020).

[329] Alejandro Baldominos, Yago Saez, and Pedro Isasi. «Hybridizing evolutionary computation and deep neural networks: an approach to handwriting recognition using committees and transfer learning.» In: *Complexity* 2019 (2019).

[330] Aritz D Martinez, Eneko Osaba, Izaskun Oregi, Iztok Fister, Iztok Fister, and Javier Del Ser. «Hybridizing differential evolution and novelty search for multimodal optimization problems.» In: *Genetic and Evolutionary Computation Conference Companion*. 2019, pp. 1980–1989.

[331] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. «Similarity of neural network representations revisited.» In: *International Conference on Machine Learning*. 2019.

[332] Aloïs Pourchot and Olivier Sigaud. «CEM-RL: Combining evolutionary and gradient-based methods for policy search.» In: *arXiv preprint arXiv:1810.01222* (2018).

[333] Krzysztof Maziarz, Mingxing Tan, Andrey Khorlin, Kuang-Yu Samuel Chang, Stanislaw Jastrzebski, Quentin de Laroussilhe, and Andrea Gesmundo. «Evolutionary-neural hybrid agents for architecture search.» In: *arXiv preprint arXiv:1811.09828* (2018).

[334] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. «What is the state of neural network pruning?» In: *arXiv preprint arXiv:2003.03033* (2020).

[335] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. «Survey of dropout methods for deep neural networks.» In: *arXiv preprint arXiv:1904.13310* (2019).

[336] Zhenyu Wang, Fu Li, Guangming Shi, Xuemei Xie, and Fangyu Wang. «Network pruning using sparse learning and genetic algorithm.» In: *Neurocomputing* (2020).

[337] James O' Neill. «An overview of neural network compression.» In: *arXiv preprint arXiv:2006.03669* (2020).

[338] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. «Deep learning for IoT big data and streaming analytics: A survey.» In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 2923–2960.

[339] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. «Federated machine learning: Concept and applications.» In: *ACM Transactions on Intelligent Systems and Technology* 10.2 (2019), pp. 1–19.

[340] Jiasi Chen and Xukan Ran. «Deep learning with edge computing: A review.» In: *IEEE* 107.8 (2019), pp. 1655–1674.

[341] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. «Estimation of energy consumption in machine learning.» In: *Journal of Parallel and Distributed Computing* 134 (2019), pp. 75–88.

[342] Milad Nasr, Reza Shokri, and Amir Houmansadr. «Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks.» In: *arXiv preprint arXiv:1812.00910* (2018).

[343] Nuria Rodríguez-Barroso, Goran Stipcich, Daniel Jiménez-López, José Antonio Ruiz-Millán, Eugenio Martínez-Cámara, Gerardo González-Seco, M Luzón, Miguel Ángel Veganzones, and Francisco Herrera. «Federated Learning and Differential Privacy: Software tools analysis, the Sherpa. ai FL framework and methodological guidelines for preserving data privacy.» In: *Information Fusion* (2020).

[344] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. «Analyzing federated learning through an adversarial lens.» In: *International Conference on Machine Learning*. 2019, pp. 634–643.

[345] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems.* Vol. 5. 2007.

[346] Efrén Mezura-Montes and Carlos A Coello Coello. «Constraint-handling in nature-inspired numerical optimization: past, present and future.» In: *Swarm and Evolutionary Computation* 1.4 (2011), pp. 173–194.

[347] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. «Insights on transfer optimization: Because experience is the best teacher.» In: *IEEE Transactions on Emerging Topics in Computing* 2.1 (2017), pp. 51–64.

[348] Liang Feng, Yew-Soon Ong, Ah-Hwee Tan, and Ivor W Tsang. «Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems.» In: *Memetic Computing* 7.3 (2015), pp. 159–180.

[349] Abhishek Gupta and Yew-Soon Ong. «Genetic transfer or population diversification? Deciphering the secret ingredients of evolutionary multitask optimization.» In: *2016 IEEE Symposium Series on Computational Intelligence*. IEEE. 2016, pp. 1–7.

[350] Yew-Soon Ong. «Towards evolutionary multitasking: a new paradigm in evolutionary computation.» In: *Computational Intelligence, Cyber Security and Computational Models*. 2016, pp. 25–26.

[351] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. *Handbook of evolutionary computation.* 1997.

[352] James Kennedy. «Swarm intelligence.» In: *Handbook of nature-inspired and innovative computing.* 2006, pp. 187–219.

[353] Maoguo Gong, Zedong Tang, Hao Li, and Jun Zhang. «Evolutionary Multitasking with Dynamic Resource Allocating Strategy.» In: *IEEE Transactions on Evolutionary Computation* 23.5 (2019), pp. 858–869.

[354] Yanan Yu, Anmin Zhu, Zexuan Zhu, Qiuzhen Lin, Jian Yin, and Xiaoliang Ma. «Multifactorial Differential Evolution with Opposition-based Learning for Multi-tasking Optimization.» In: *IEEE Congress on Evolutionary Computation.* 2019, pp. 1898–1905.

[355] Kevin Swersky, Jasper Snoek, and Ryan P Adams. «Multi-task bayesian optimization.» In: *Advances in Neural Information Processing Systems* 26 (2013), pp. 2004–2012.

[356] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. «Taking the human out of the loop: A review of Bayesian optimization.» In: *IEEE* 104.1 (2015), pp. 148–175.

[357] Henry B Moss, David S Leslie, and Paul Rayson. «Mumbo: Multi-task max-value bayesian optimization.» In: *arXiv preprint arXiv:2006.12093* (2020).

[358] Michael Pearce and Juergen Branke. «Continuous multi-task Bayesian Optimisation with correlation.» In: *European Journal of Operational Research* 270.3 (2018), pp. 1074–1085.

[359] Sayak Ray Chowdhury and Aditya Gopalan. «No-regret Algorithms for Multi-task Bayesian Optimization.» In: *arXiv preprint arXiv:2008.08885* (2020).

[360] Kay Chen Tan, Liang Feng, and Min Jiang. «Evolutionary Transfer Optimization-A New Frontier in Evolutionary Computation Research.» In: *IEEE Computational Intelligence Magazine* 16.1 (2021), pp. 22–33.

[361] Qingzheng Xu, Na Wang, Lei Wang, Wei Li, and Qian Sun. «Multi-Task Optimization and Multi-Task Evolutionary Computation in the Past Five Years: A Brief Review.» In: *Mathematics* 9.8 (2021), p. 864.

[362] Tingyang Wei, Shibin Wang, Jinghui Zhong, Dong Liu, and Jun Zhang. «A Review on Evolutionary Multi-Task Optimization: Trends and Challenges.» In: *IEEE Transactions on Evolutionary Computation* (2021).

[363] Kavitesh Kumar Bali, Abhishek Gupta, Liang Feng, Yew Soon Ong, and Tan Puay Siew. «Linearized domain adaptation in evolutionary multitasking.» In: *2017 IEEE Congress on Evolutionary Computation.* IEEE. 2017, pp. 1295–1302.

[364] Sushil J Louis and John McDonnell. *Learning with case-injected genetic algorithms.* Tech. rep. College of Engineering, University of Nevada, Reno, 2004.

[365] Bingshui Da, Yew-Soon Ong, Liang Feng, A Kai Qin, Abhishek Gupta, Zexuan Zhu, Chuan-Kang Ting, Ke Tang, and Xin Yao. «Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results.» In: *arXiv preprint arXiv:1706.03470* (2017).

[366] Jan Paredis. «Coevolutionary computation.» In: *Artificial Life* 2.4 (1995), pp. 355–375.

[367] Hui Song, AK Qin, Pei-Wei Tsai, and JJ Liang. «Multitasking Multi-Swarm Optimization.» In: *IEEE Congress on Evolutionary Computation.* 2019, pp. 1937–1944.

[368] Mei-Ying Cheng, Abhishek Gupta, Yew-Soon Ong, and Zhi-Wei Ni. «Coevolutionary multitasking for concurrent global optimization: With case studies in complex engineering design.» In: *Engineering Applications of Artificial Intelligence* 64 (2017), pp. 13–24.

[369] Eneko Osaba, Esther Villar-Rodriguez, and Javier Del Ser. «A Coevolutionary Variable Neighborhood Search Algorithm for Discrete Multitasking (CoVNS): Application to Community Detection over Graphs.» In: *2020 IEEE Symposium Series on Computational Intelligence*. IEEE. 2020, pp. 768–774.

[370] Eneko Osaba, Javier Del Ser, Aritz D Martinez, Jesus L Lobo, and Francisco Herrera. «AT-MFCGA: An Adaptive Transfer-guided Multifactorial Cellular Genetic Algorithm for Evolutionary Multitasking.» In: *arXiv:2010.03917* (2020).

[371] Eneko Osaba, Aritz D Martinez, Jesus L Lobo, Javier Del Ser, and Francisco Herrera. «Multifactorial Cellular Genetic Algorithm (MFCGA): Algorithmic Design, Performance Comparison and Genetic Transferability Analysis.» In: *arXiv:2003.10768* (2020).

[372] Pedro Larrañaga and Jose A Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*. Vol. 2. 2001.

[373] Abhishek Gupta, Yew-Soon Ong, Bingshui Da, Liang Feng, and Stephanus Daniel Handoko. «Landscape synergy in evolutionary multitasking.» In: *2016 IEEE Congress on Evolutionary Computation*. IEEE. 2016, pp. 3076–3083.

[374] Kavitesh Kumar Bali, Yew-Soon Ong, Abhishek Gupta, and Puay Siew Tan. «Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II.» In: *IEEE Transactions on Evolutionary Computation* 24.1 (2019), pp. 69–83.

[375] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. «Distributed evolutionary algorithms and their models: A survey of the state-of-the-art.» In: *Applied Soft Computing* 34 (2015), pp. 286–300.

[376] Yew Soon Ong. «Towards evolutionary multitasking: a new paradigm.» In: *Symposium on Information and Communication Technology*. 2015, pp. 2–2.

[377] A Gupta, YS Ong, B Da, L Feng, and S Handoko. «Measuring complementarity between function landscapes in evolutionary multitasking.» In: *2016 IEEE Congress on Evolutionary Computation, accepted*. 2016.

[378] Abhishek Gupta, Bingshui Da, Yuan Yuan, and Yew-Soon Ong. «On the emerging notion of evolutionary multitasking: A computational analog of cognitive multitasking.» In: *Recent Advances in Evolutionary Multi-objective Optimization*. 2017, pp. 139–157.

[379] Zhengxin Huang, Zefeng Chen, and Yuren Zhou. «Analysis on the Efficiency of Multifactorial Evolutionary Algorithms.» In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2020, pp. 634–647.

[380] Qingzheng Xu, Jianhang Zhang, Rong Fei, and Wei Li. «Parameter analysis on multi-factorial evolutionary algorithm.» In: *The Journal of Engineering* 2020.13 (2020), pp. 620–625.

[381] Na Wang, Qingzheng Xu, Rong Fei, Jungang Yang, and Lei Wang. «Rigorous Analysis of Multi-Factorial Evolutionary Algorithm as Multi-Population Evolution Model.» In: *International Journal of Computational Intelligence Systems* 12.2 (2019), pp. 1121–1133.

[382] Ryuichi Hashimoto, Hisao Ishibuchi, Naoki Masuyama, and Yusuke Nojima. «Analysis of evolutionary multi-tasking as an island model.» In: *Genetic and Evolutionary Computation Conference Companion*. 2018, pp. 1894–1897.

[383] Lei Zhou, Liang Feng, Jinghui Zhong, Zexuan Zhu, Bingshui Da, and Zhou Wu. «A study of similarity measure between tasks for multifactorial evolutionary algorithm.» In: *Genetic and Evolutionary Computation Conference Companion*. 2018, pp. 229–230.

[384] Lizhong Yao, Wei Long, Jun Yi, Taifu Li, Dedong Tang, and Qingzheng Xu. «A novel tournament selection based on multilayer cultural characteristics in gene-culture coevolutionary multitasking.» In: *Soft Computing* 25.14 (2021), pp. 9529–9543.

[385] Lei Wang, Qian Sun, Qingzheng Xu, Wei Li, and Qiaoyong Jiang. «Analysis of Multitasking Evolutionary Algorithms under the Order of Solution Variables.» In: *Complexity* 2020 (2020).

[386] Lu Bai, Wu Lin, Abhishek Gupta, and Yew-Soon Ong. «From Multitask Gradient Descent to Gradient-Free Evolutionary Multitasking: A Proof of Faster Convergence.» In: *IEEE Transactions on Cybernetics* (2021).

[387] Abhishek Gupta and Yew-Soon Ong. «Back to the roots: Multi-X evolutionary computation.» In: *Cognitive Computation* 11.1 (2019), pp. 1–17.

[388] Genghui Li, Qingfu Zhang, and Zhenkun Wang. «Evolutionary Competitive Multitasking Optimization.» In: *IEEE Transactions on Evolutionary Computation* (2022).

[389] Yuan Yuan, Yew-Soon Ong, Liang Feng, A Kai Qin, Abhishek Gupta, Bingshui Da, Qingfu Zhang, Kay Chen Tan, Yaochu Jin, and Hisao Ishibuchi. «Evolutionary multitasking for multiobjective continuous optimization: Benchmark problems, performance metrics and baseline results.» In: *arXiv preprint arXiv:1706.02766* (2017).

[390] Yuan Yuan, Yew-Soon Ong, Abhishek Gupta, Puay Siew Tan, and Hua Xu. «Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP.» In: *2016 IEEE Region 10 Conference*. IEEE. 2016, pp. 3157–3164.

[391] Lei Zhou, Liang Feng, Jinghui Zhong, Yew-Soon Ong, Zexuan Zhu, and Edwin Sha. «Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem.» In: *2016 IEEE Symposium Series on Computational Intelligence*. IEEE. 2016, pp. 1–8.

[392] Pham Dinh Thanh, Huynh Thi Thanh Binh, and Tran Ba Trung. «An efficient strategy for using multifactorial optimization to solve the clustered shortest path tree problem.» In: *Applied Intelligence* 50.4 (2020), pp. 1233–1258.

[393] Huynh ThiThanh Binh, Pham Dinh Thanh, Tran Ba Trung, et al. «Effective multifactorial evolutionary algorithm for solving the cluster shortest path tree problem.» In: *2018 IEEE Congress on Evolutionary Computation*. IEEE. 2018, pp. 1–8.

[394] Pham Dinh Thanh, Dinh Anh Dung, Tran Ngoc Tien, and Huynh Thi Thanh Binh. «An effective representation scheme in multifactorial evolutionary algorithm for solving cluster shortest-path tree problem.» In: *2018 IEEE Congress on Evolutionary Computation*. IEEE. 2018, pp. 1–8.

[395] Binh Huynh Thi Thanh and Thanh Pham Dinh. «Two levels approach based on multifactorial optimization to solve the clustered shortest path tree problem.» In: *Evolutionary Intelligence* (2020), pp. 1–29.

[396] Thanh Pham Dinh, Binh Huynh Thi Thanh, Trung Tran Ba, and Long Nguyen Binh. «Multifactorial evolutionary algorithm for solving clustered tree problems: competition among cayley codes.» In: *Memetic Computing* 12.3 (2020), pp. 185–217.

[397] Phan Thi Hong Hanh, Pham Dinh Thanh, and Huynh Thi Thanh Binh. «Evolutionary algorithm and multifactorial evolutionary algorithm on clustered shortest-path tree problem.» In: *Information Sciences* 553 (2021), pp. 280–304.

[398] Huynh Thi Thanh Binh, Ta Bao Thang, Nguyen Duc Thai, and Pham Dinh Thanh. «A bi-level encoding scheme for the clustered shortest-path tree problem in multifactorial optimization.» In: *Engineering Applications of Artificial Intelligence* 100 (2021), p. 104187.

[399] Liang Bao, Yutao Qi, Mengqing Shen, Xiaoxuan Bu, Jusheng Yu, Qian Li, and Ping Chen. «An evolutionary multitasking algorithm for cloud computing service composition.» In: *World Congress on Services*. Springer. 2018, pp. 130–144.

[400] Chen Wang, Hui Ma, Gang Chen, and Sven Hartmann. «Evolutionary multitasking for semantic web service composition.» In: *2019 IEEE Congress on Evolutionary Computation*. IEEE. 2019, pp. 2490–2497.

[401] Zhengping Liang, Jian Zhang, Liang Feng, and Zexuan Zhu. «Multifactorial Optimization for Large-scale Virtual Machine Placement in Cloud Computing.» In: *arXiv preprint arXiv:2001.06585* (2020).

[402] Jinliang Ding, Cuie Yang, Yaochu Jin, and Tianyou Chai. «Generalized multitasking for evolutionary optimization of expensive problems.» In: *IEEE Transactions on Evolutionary Computation* 23.1 (2017), pp. 44–58.

[403] Jian Yin, Anmin Zhu, Zexuan Zhu, Yanan Yu, and Xiaoliang Ma. «Multifactorial evolutionary algorithm enhanced with cross-task search direction.» In: *2019 IEEE Congress on Evolutionary Computation*. IEEE. 2019, pp. 2244–2251.

[404] Thi Thanh Binh Huynh, Dinh Thanh Pham, Ba Trung Tran, Cong Thanh Le, Minh Hai Phong Le, Ananthram Swami, and Thu Lam Bui. «A multifactorial optimization paradigm for linkage tree genetic algorithm.» In: *Information Sciences* 540 (2020), pp. 325–344.

[405] Bingshui Da, Abhishek Gupta, Yew Soon Ong, and Liang Feng. «The boon of gene-culture interaction for effective evolutionary multitasking.» In: *Australasian Conference on Artificial Life and Computational Intelligence*. Springer. 2016, pp. 54–65.

[406] Yuchen Lian, Zhengxin Huang, Yuren Zhou, and Zefeng Chen. «Improve theoretical upper bound of jumpk function by evolutionary multitasking.» In: *High Performance Computing and Cluster Technologies Conference*. 2019, pp. 44–50.

[407] Yongjian Zhou, Tonghao Wang, and Xingguang Peng. «MFEA-IG: A multi-task algorithm for mobile agents path planning.» In: *IEEE Congress on Evolutionary Computation*. 2020, pp. 1–7.

[408] Huynh Thi Thanh Binh, Ta Bao Thangy, Nguyen Binh Long, Ngo Viet Hoang, and Pham Dinh Thanh. «Multifactorial Evolutionary Algorithm for Inter-Domain Path Computation under Domain Uniqueness Constraint.» In: *2020 IEEE Congress on Evolutionary Computation*. IEEE. 2020, pp. 1–8.

[409] Tran Cong Dao, Tran Huy Hung, Nguyen Thi Tam, and Huynh Thi Thanh Binh. «A Multifactorial Evolutionary Algorithm For Minimum Energy Cost Data Aggregation Tree In Wireless Sensor Networks.» In: *2021 IEEE Congress on Evolutionary Computation*. IEEE. 2021, pp. 1656–1663.

[410] Nguyen Thi Tam, Vi Thanh Dat, Phan Ngoc Lan, Huynh Thi Thanh Binh, Ananthram Swami, et al. «Multifactorial evolutionary optimization to maximize lifetime of wireless sensor network.» In: *Information Sciences* (2021).

[411] Ting-Chen Wang and Rung-Tzuo Liaw. «Multifactorial genetic fuzzy data mining for building membership functions.» In: *2020 IEEE Congress on Evolutionary Computation*. IEEE. 2020, pp. 1–8.

[412] Xiaoming Xue, Kai Zhang, Kay Chen Tan, Liang Feng, Jian Wang, Guodong Chen, Xinggang Zhao, Liming Zhang, and Jun Yao. «Affine Transformation-Enhanced Multifactorial Optimization for Heterogeneous Problems.» In: *IEEE Transactions on Cybernetics* (2020).

[413] Amit Rauniyar, Rahul Nath, and Pranab K Muhuri. «Multi-factorial evolutionary algorithm based novel solution approach for multi-objective pollution-routing problem.» In: *Computers & Industrial Engineering* 130 (2019), pp. 757–771.

[414] Junwei Liu, Peiling Li, Guibin Wang, Yongxing Zha, Jianchun Peng, and Gang Xu. «A Multitasking Electric Power Dispatch Approach With Multi-Objective Multifactorial Optimization Algorithm.» In: *IEEE Access* 8 (2020), pp. 155902–155911.

[415] Cuie Yang, Jinliang Ding, Yaochu Jin, Chengzhi Wang, and Tianyou Chai. «Multitasking multiobjective evolutionary operational indices optimization of beneficiation processes.» In: *IEEE Transactions on Automation Science and Engineering* 16.3 (2018), pp. 1046–1057.

[416] Cuie Yang, Jinliang Ding, Kay Chen Tan, and Yaochu Jin. «Two-stage assortative mating for multi-objective multifactorial evolutionary optimization.» In: *2017 IEEE 56th Annual Conference on Decision and Control*. IEEE. 2017, pp. 76–81.

[417] Jiajie Mo, Zhun Fan, Wenji Li, Yi Fang, Yugen You, and Xinye Cai. «Multi-factorial evolutionary algorithm based on M2M decomposition.» In: *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer. 2017, pp. 134–144.

[418] Zifeng Zhou, Xiaoliang Ma, Zhengping Liang, and Zexuan Zhu. «Multi-objective multi-factorial memetic algorithm based on bone route and large neighborhood local search for VRPTW.» In: *2020 IEEE Congress on Evolutionary Computation*. IEEE. 2020, pp. 1–8.

[419] Nguyen Quoc Tuan, Ta Duy Hoang, and Huynh Thi Thanh Binh. «A guided differential evolutionary multi-tasking with powell search method for solving multi-objective continuous optimization.» In: *2018 IEEE Congress on Evolutionary Computation*. IEEE. 2018, pp. 1–8.

[420] Jun Yi, Junren Bai, Haibo He, Wei Zhou, and Lizhong Yao. «A Multifactorial Evolutionary Algorithm for Multitasking Under Interval Uncertainties.» In: *IEEE Transactions on Evolutionary Computation* (2020).

[421] Jun Yi, Wei Zhang, Junren Bai, Wei Zhou, and Lizhong Yao. «Multifactorial Evolutionary Algorithm Based on Improved Dynamical Decomposition for Many-objective Optimization Problems.» In: *IEEE Transactions on Evolutionary Computation* (2021).

[422] Qiuhua Tang, Kai Meng, Lixin Cheng, and Zikai Zhang. «An improved multi-objective multifactorial evolutionary algorithm for assembly line balancing problem considering regular production and preventive maintenance scenarios.» In: *Swarm and Evolutionary Computation* 68 (2022), p. 101021.

[423] Bingshui Da, Abhishek Gupta, Yew-Soon Ong, and Liang Feng. «Evolutionary multitasking across single and multi-objective formulations for improved problem solving.» In: *2016 IEEE Congress on Evolutionary Computation*. IEEE. 2016, pp. 1695–1701.

[424] Xiaoliang Ma, Jian Yin, Anmin Zhu, Xiaodong Li, Yanan Yu, Lei Wang, Yutao Qi, and Zexuan Zhu. «Enhanced Multifactorial Evolutionary Algorithm With Meme Helper-Tasks.» In: *IEEE Transactions on Cybernetics* (2021).

[425] Abhishek Gupta, Jacek Mańdziuk, and Yew-Soon Ong. «Evolutionary multitasking in bi-level optimization.» In: *Complex & Intelligent Systems* 1.1-4 (2015), pp. 83–95.

[426] Ramon Sagarna and Yew-Soon Ong. «Concurrently searching branches in software tests generation through multitask evolution.» In: *2016 IEEE Symposium Series on Computational Intelligence*. IEEE. 2016, pp. 1–8.

[427] L Feng, W Zhou, L Zhou, SW Jiang, JH Zhong, BS Da, ZX Zhu, and Y Wang. «An empirical study of multifactorial PSO and multifactorial DE.» In: *2017 IEEE Congress on Evolutionary Computation*. IEEE. 2017, pp. 921–928.

[428] Kenneth Price, Rainer M Storn, and Jouni A Lampinen. *Differential evolution: a practical approach to global optimization*. 2006.

[429] Lei Zhou, Liang Feng, Kai Liu, Chao Chen, Shaojiang Deng, Tao Xiang, and Siwei Jiang. «Towards effective mutation for knowledge transfer in multifactorial differential evolution.» In: *2019 IEEE Congress on Evolutionary Computation*. IEEE. 2019, pp. 1541–1547.

[430] Zedong Tang, Maoguo Gong, Yue Wu, Wenfeng Liu, and Yu Xie. «Regularized Evolutionary Multi-Task Optimization: Learning to Inter-Task Transfer in Aligned Subspace.» In: *IEEE Transactions on Evolutionary Computation* (2020).

[431] Ke Chen, Bing Xue, Mengjie Zhang, and Fengyu Zhou. «An Evolutionary Multitasking-Based Feature Selection Method for High-Dimensional Classification.» In: *IEEE Transactions on Cybernetics* (2020).

[432] Eneko Osaba, Aritz D Martinez, Jesus L Lobo, Ibai Laña, and Javier Del Ser. «On the Transferability of Knowledge among Vehicle Routing Problems by using a Cellular Evolutionary Multitasking.» In: *arXiv preprint arXiv:2005.05066* (2020).

[433] Heng Xiao, Gen Yokoya, and Toshiharu Hatanaka. «Multifactorial pso-fa hybrid algorithm for multiple car design benchmark.» In: *2019 IEEE International Conference on Systems, Man and Cybernetics*. IEEE. 2019, pp. 1926–1931.

[434] Yinglan Feng, Liang Feng, Yaqing Hou, and Kay Chen Tan. «Large-Scale optimization via Evolutionary Multitasking assisted Random Embedding.» In: *IEEE Congress on Evolutionary Computation*. 2020, pp. 1–8.

[435] Zhengping Liang, Jian Zhang, Liang Feng, and Zexuan Zhu. «A hybrid of genetic transform and hyper-rectangle search strategies for evolutionary multi-tasking.» In: *Expert Systems with Applications* 138 (2019), p. 112798.

[436] Xingxing Hao, Rong Qu, and Jing Liu. «A Unified Framework of Graph-based Evolutionary Multitasking Hyper-heuristic.» In: *IEEE Transactions on Evolutionary Computation* (2020).

[437] Wei Guo, Feng Zou, Debao Chen, Hui Liu, and Siyu Cao. «An Improved Teaching-Learning-Based Optimization for Multitask Optimization Problems.» In: *International Conference on Intelligent Computing*. Springer. 2021, pp. 48–58.

[438] Chao Wang, Jing Liu, Kai Wu, and Chaolong Ying. «Learning large-scale fuzzy cognitive maps using an evolutionary many-task algorithm.» In: *Applied Soft Computing* 108 (2021), p. 107441.

[439] Genghui Li, Qingfu Zhang, and Weifeng Gao. «Multipopulation evolution framework for multifactorial optimization.» In: *Genetic and Evolutionary Computation Conference Companion*. 2018, pp. 215–216.

[440] Genghui Li, Qiuzhen Lin, and Weifeng Gao. «Multifactorial optimization via explicit multipopulation evolutionary framework.» In: *Information Sciences* 512 (2020), pp. 1555–1570.

[441] Jinghui Zhong, Liang Feng, Wentong Cai, and Yew-Soon Ong. «Multifactorial genetic programming for symbolic regression problems.» In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2018).

[442] Wei Li and Jinbo Li. «Covariance Matrix Adaptation Evolutionary Algorithm for Multi-task Optimization.» In: *Bio-Inspired Computing: Theories and Applications: 15th International Conference, BIC-TA 2020, Qingdao, China, October 23-25, 2020, Revised Selected Papers*. Vol. 1363. Springer Nature. 2021, p. 25.

[443] Fang Shen, Jing Liu, and Kai Wu. «Evolutionary multitasking fuzzy cognitive map learning.» In: *Knowledge-Based Systems* 192 (2020), p. 105294.

[444] Hao Li, Yew-Soon Ong, Maoguo Gong, and Zhenkun Wang. «Evolutionary multitasking sparse reconstruction: Framework and case study.» In: *IEEE Transactions on Evolutionary Computation* 23.5 (2018), pp. 733–747.

[445] Chen Jin, Pei-Wei Tsai, and A Kai Qin. «A Study on Knowledge Reuse Strategies in Multitasking Differential Evolution.» In: *2019 IEEE Congress on Evolutionary Computation*. IEEE. 2019, pp. 1564–1571.

[446] Zhiwei Xu, Kai Zhang, Xin Xu, and Juanjuan He. «A Fireworks Algorithm Based on Transfer Spark for Evolutionary Multitasking.» In: *Frontiers in Neurorobotics* 13 (2020), p. 109.

[447] Ying Tan and Yuanchun Zhu. «Fireworks algorithm for optimization.» In: *International conference in swarm intelligence*. Springer. 2010, pp. 355–364.

[448] Fangfang Zhang, Yi Mei, Su Nguyen, and Mengjie Zhang. «A preliminary approach to evolutionary multitasking for dynamic flexible job shop scheduling via genetic programming.» In: *Genetic and Evolutionary Computation Conference Companion*. 2020, pp. 107–108.

[449] Xianpeng Wang, Zhiming Dong, Lixin Tang, and Qingfu Zhang. «Multiobjective Multitasking Optimization Based on Decomposition with Dual Neighborhoods.» In: *arXiv preprint arXiv:2101.07548* (2021).

[450] Qingfu Zhang and Hui Li. «MOEA/D: A multiobjective evolutionary algorithm based on decomposition.» In: *IEEE Transactions on Evolutionary Computation* 11.6 (2007), pp. 712–731.

[451] Qingzheng Xu, Lei Wang, Jungang Yang, Na Wang, Rong Fei, and Qian Sun. «An Effective Variable Transformation Strategy in Multitasking Evolutionary Algorithms.» In: *Complexity* 2020 (2020).

[452] Jing Liang, Kangjia Qiao, Minghua Yuan, Kunjie Yu, Boyang Qu, Shilei Ge, Yaxin Li, and Guanlin Chen. «Evolutionary multi-task optimization for parameters extraction of photovoltaic models.» In: *Energy Conversion and Management* 207 (2020), p. 112509.

[453] Eneko Osaba, Aritz D Martinez, Akemi Galvez, Andres Iglesias, and Javier Del Ser. «dMFEA-II: An Adaptive Multifactorial Evolutionary Algorithm for Permutation-based Discrete Optimization Problems.» In: *arXiv:2004.06559* (2020).

[454] Xiaolong Zheng, A Kai Qin, Maoguo Gong, and Deyun Zhou. «Self-regulated evolutionary multitask optimization.» In: *IEEE Transactions on Evolutionary Computation* 24.1 (2019), pp. 16–28.

[455] Yu-Wei Wen and Chuan-Kang Ting. «Parting ways and reallocating resources in evolutionary multitasking.» In: *2017 IEEE Congress on Evolutionary Computation*. IEEE. 2017, pp. 2404–2411.

[456] Ting Yee Lim, Choo Jun Tan, Wai Peng Wong, and Chee Peng Lim. «An information entropy-based evolutionary computation for multi-factorial optimization.» In: *Applied Soft Computing* 114 (2022), p. 108071.

[457] Jing Tang, Yingke Chen, Zixuan Deng, Yanping Xiang, and Colin Paul Joy. «A Group-based Approach to Improve Multifactorial Evolutionary Algorithm.» In: *IJCAI*. 2018, pp. 3870–3876.

[458] Lei Zhou, Liang Feng, Kay Chen Tan, Jinghui Zhong, Zexuan Zhu, Kai Liu, and Chao Chen. «Toward Adaptive Knowledge Transfer in Multifactorial Evolutionary Computation.» In: *IEEE Transactions on Cybernetics* (2020).

[459] Jun Yao, Yandong Nie, Zihao Zhao, Xiaoming Xue, Kai Zhang, Chuanjin Yao, Liming Zhang, Jian Wang, and Yongfei Yang. «Self-adaptive multifactorial evolutionary algorithm for multitasking production optimization.» In: *Journal of Petroleum Science and Engineering* 205 (2021), p. 108900.

[460] Huynh Thi Thanh Binh, Nguyen Quoc Tuan, and Doan Cao Thanh Long. «A multi-objective multi-factorial evolutionary algorithm with reference-point-based approach.» In: *2019 IEEE Congress on Evolutionary Computation*. IEEE. 2019, pp. 2824–2831.

[461] Ting Wu, Siqi Bu, Xiang Wei, Guibin Wang, and Bin Zhou. «Multitasking multi-objective operation optimization of integrated energy system considering biogas-solar-wind renewables.» In: *Energy Conversion and Management* 229 (2021), p. 113736.

[462] Qunjian Chen, Xiaoliang Ma, Yiwen Sun, and Zexuan Zhu. «Adaptive memetic algorithm based evolutionary multi-tasking single-objective optimization.» In: *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer. 2017, pp. 462–472.

[463] Zedong Tang and Maoguo Gong. «Adaptive multifactorial particle swarm optimisation.» In: *CEvol. Comput.AAI Transactions on Intelligence Technology* 4.1 (2019), pp. 37–46.

[464] Zedong Tang, Maoguo Gong, Yu Xie, Hao Li, and AK Qin. «Multi-Task Particle Swarm Optimization With Dynamic Neighbor and Level-Based Inter-Task Learning.» In: *IEEE Transactions on Emerging Topics in Computational Intelligence* (2021).

[465] Aritz D Martinez, Javier Del Ser, Eneko Osaba, and Francisco Herrera. «Adaptive Multi-factorial Evolutionary Optimization for Multi-task Reinforcement Learning.» In: *IEEE Transactions on Evolutionary Computation* (2021).

[466] Zan Wang and Xianpeng Wang. «Multiobjective multifactorial operation optimization for continuous annealing production process.» In: *Industrial & Engineering Chemistry Research* 58.41 (2019), pp. 19166–19178.

[467] Zhengping Liang, Hao Dong, Cheng Liu, Weiqi Liang, and Zexuan Zhu. «Evolutionary multitasking for multiobjective optimization with subspace alignment and adaptive differential evolution.» In: *IEEE Transactions on Cybernetics* (2020).

[468] Zhiwei Xu, Xiaoming Liu, Kai Zhang, and Juanjuan He. «Cultural transmission based multi-objective evolution strategy for evolutionary multitasking.» In: *Information Sciences* 582 (2022), pp. 215–242.

[469] Yizhe Zhao, Hao Li, Yue Wu, Shanfeng Wang, and Maoguo Gong. «Endmember selection of hyperspectral images based on evolutionary multitask.» In: *2020 IEEE Congress on Evolutionary Computation*. IEEE. 2020, pp. 1–7.

[470] Rung-Tzuo Liaw and Chuan-Kang Ting. «Evolutionary many-tasking based on biocoenosis through symbiosis: A framework and benchmark problems.» In: *2017 IEEE Congress on Evolutionary Computation*. IEEE. 2017, pp. 2266–2273.

[471] Rung-Tzuo Liaw and Chuan-Kang Ting. «Evolutionary manytasking optimization based on symbiosis in biocoenosis.» In: *AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 4295–4303.

[472] Rung-Tzuo Liaw and Chuan-Kang Ting. «Evolution of biocoenosis through symbiosis with fitness approximation for many-tasking optimization.» In: *Memetic Computing* 12.4 (2020), pp. 399–417.

[473] Ying Bi, Bing Xue, and Mengjie Zhang. «Learning to Share: A Multitasking Genetic Programming Approach to Image Feature Learning.» In: *arXiv e-prints* (2020), arXiv–2012.

[474] Qianlong Dang, Weifeng Gao, and Maoguo Gong. «Multiobjective multitasking optimization assisted by multidirectional prediction method.» In: *Complex & Intelligent Systems* (2022), pp. 1–17.

[475] Yongliang Chen, Jinghui Zhong, Liang Feng, and Jun Zhang. «An adaptive archive-based evolutionary framework for many-task optimization.» In: *IEEE Transactions on Emerging Topics in Computational Intelligence* (2019).

[476] Eneko Osaba, Javier Del Ser, Xin-She Yang, Andres Iglesias, and Akemi Galvez. «COEBA: A Coevolutionary Bat Algorithm for Discrete Evolutionary Multitasking.» In: *arXiv preprint arXiv:2003.11628* (2020).

[477] Xiaolong Zheng, Yu Lei, A Kai Qin, Deyun Zhou, Jiao Shi, and Maoguo Gong. «Differential evolutionary multi-task optimization.» In: *2019 IEEE Congress on Evolutionary Computation*. IEEE. 2019, pp. 1914–1921.

[478] Dongrui Wu and Xianfeng Tan. «Multitasking Genetic Algorithm (MTGA) for Fuzzy System Optimization.» In: *IEEE Transactions on Fuzzy Systems* 28.6 (2020), pp. 1050–1061.

[479] Jiao Shi, Xi Zhang, Xiaodong Liu, Yu Lei, and Gwanggil Jeon. «Multicriteria semi-supervised hyperspectral band selection based on evolutionary multitask optimization.» In: *Knowledge-Based Systems* (2022), p. 107934.

[480] Liang Feng, Lei Zhou, Jinghui Zhong, Abhishek Gupta, Yew-Soon Ong, Kay-Chen Tan, and Alex Kai Qin. «Evolutionary multitasking via explicit autoencoding.» In: *IEEE Transactions on Cybernetics* 49.9 (2018), pp. 3457–3470.

[481] Liang Feng, Yuxiao Huang, Lei Zhou, Jinghui Zhong, Abhishek Gupta, Ke Tang, and Kay Chen Tan. «Explicit Evolutionary Multitasking for Combinatorial Optimization: A Case Study on Capacitated Vehicle Routing Problem.» In: *IEEE Transactions on Cybernetics* (2020).

[482] Jiabin Lin, Hai-Lin Liu, Kay Chen Tan, and Fangqing Gu. «An Effective Knowledge Transfer Approach for Multiobjective Multitasking Optimization.» In: *IEEE Transactions on Cybernetics* (2020).

[483] Zedong Tang, Maoguo Gong, Fenlong Jiang, Hao Li, and Yue Wu. «Multipopulation optimization for multitask optimization.» In: *2019 IEEE Congress on Evolutionary Computation.* IEEE. 2019, pp. 1906–1913.

[484] Ke Zhang, Wen-Ning Hao, Xiao-Han Yu, Da-Wei Jin, and Zhong-Hui Zhang. «A Multitasking Genetic Algorithm for Mamdani Fuzzy System with Fully Overlapping Triangle Membership Functions.» In: *International Journal of Fuzzy Systems* (2020), pp. 1–17.

[485] Dingnan Liu, Shijia Huang, and Jinghui Zhong. «Surrogate-assisted multi-tasking memetic algorithm.» In: *2018 IEEE Congress on Evolutionary Computation.* IEEE. 2018, pp. 1–8.

[486] Handing Wang, Liang Feng, Yaochu Jin, and John Doherty. «Surrogate-Assisted Evolutionary Multitasking for Expensive Minimax Optimization in Multiple Scenarios.» In: *IEEE Computational Intelligence Magazine* 16.1 (2021), pp. 34–48.

[487] Fangfang Zhang, Yi Mei, Su Nguyen, Mengjie Zhang, and Kay Chen Tan. «Surrogate-Assisted Evolutionary Multitask Genetic Programming for Dynamic Flexible Job Shop Scheduling.» In: *IEEE Transactions on Evolutionary Computation* (2021).

[488] Yongliang Chen, Jinghui Zhong, and Mingkui Tan. «A fast memetic multi-objective differential evolution for multi-tasking optimization.» In: *2018 IEEE Congress on Evolutionary Computation.* IEEE. 2018, pp. 1–8.

[489] Qingxia Shang, Yuxiao Huang, Yu Wang, Min Li, and Liang Feng. «Solving vehicle routing problem by memetic search with evolutionary multitasking.» In: *Memetic Computing* (2022), pp. 1–14.

[490] Zhiwei Xu and Kai Zhang. «Multiobjective multifactorial immune algorithm for multiobjective multitask optimization problems.» In: *Applied Soft Computing* 107 (2021), p. 107399.

[491] Q Shang, L Zhang, L Feng, Y Hou, J Zhong, A Gupta, Kay Chen Tan, and H-L Liu. «A preliminary study of adaptive task selection in explicit evolutionary many-tasking.» In: *2019 IEEE Congress on Evolutionary Computation*. IEEE. 2019, pp. 2153–2159.

[492] Bingshui Da, Abhishek Gupta, and Yew-Soon Ong. «Curbing negative influences online for seamless transfer evolutionary optimization.» In: *IEEE Transactions on Cybernetics* 49.12 (2018), pp. 4365–4378.

[493] Ray Lim, Lei Zhou, Abhishek Gupta, Yew-Soon Ong, and Allan N Zhang. «Solution Representation Learning in Multi-Objective Transfer Evolutionary Optimization.» In: *IEEE Access* 9 (2021), pp. 41844–41860.

[494] Liang Feng, Yew-Soon Ong, Siwei Jiang, and Abhishek Gupta. «Autoencoding evolutionary search with learning across heterogeneous problems.» In: *IEEE Transactions on Evolutionary Computation* 21.5 (2017), pp. 760–772.

[495] Chao Wang, Jing Liu, Kai Wu, and Zhaoyang Wu. «Solving Multitask Optimization Problems with Adaptive Knowledge Transfer via Anomaly Detection.» In: *IEEE Transactions on Evolutionary Computation* (2021).

[496] Abhishek Gupta and Yew-Soon Ong. «Multitask Knowledge Transfer Across Problems.» In: *Memetic Computation*. 2019, pp. 83–92.

[497] Xiaopeng Wei. «A Study on Realtime Task Selection Based on Credit Information Updating in Evolutionary Multitasking.» In: *Conference on Evolutionary Multi-Criterion Optimization*. Springer Nature. 2021, p. 480.

[498] Abhishek Gupta, Lei Zhou, Yew-Soon Ong, Zefeng Chen, and Yaqing Hou. «Half a Dozen Real-World Applications of Evolutionary Multitasking and More.» In: *arXiv preprint arXiv:2109.13101* (2021).

[499] James C Bean. «Genetic algorithms and random keys for sequencing and optimization.» In: *ORSA Journal on Computing* 6.2 (1994), pp. 154–160.

[500] Mojtaba Shakeri, Erfan Miahi, Abhishek Gupta, and Yew-Soon Ong. «Scalable Transfer Evolutionary Optimization: Coping with Big Task Instances.» In: *arXiv preprint arXiv:2012.01830* (2020).

[501] Chris D Frith. «Social cognition.» In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 363.1499 (2008), pp. 2033–2039.

[502] Roger B Myerson. *Game theory*. 2013.

[503] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. «A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms.» In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 3–18.

[504] Aritz D Martinez, Javier Del Ser, Esther Villar-Rodriguez, Eneko Osaba, Javier Poyatos, Siham Tabik, Daniel Molina, and Francisco Herrera. «Lights and shadows in Evolutionary Deep Learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges.» In: *Information Fusion* 67 (2021), pp. 161–194.

[505] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. «Evolution strategies as a scalable alternative to reinforcement learning.» In: *arXiv preprint arXiv:1703.03864* (2017).

[506] Hitoshi Iba and Nasimul Noman. *Deep Neural Evolution: Deep Learning with Evolutionary Computation.* 2020.

[507] J-B Mouret and Stéphane Doncieux. «Encouraging behavioral diversity in evolutionary robotics: An empirical study.» In: *Evolutionary Computation* 20.1 (2012), pp. 91–133.

[508] Yujin Tang, Duong Nguyen, and David Ha. «Neuroevolution of self-interpretable agents.» In: *Genetic and Evolutionary Computation Conference.* 2020, pp. 414–424.

[509] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. «Regularized evolution for image classifier architecture search.» In: *AAAI Conference on Artificial Intelligence.* Vol. 33. 2019, pp. 4780–4789.

[510] Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. «Designing neural networks through neuroevolution.» In: *Nature Machine Intelligence* 1.1 (2019), pp. 24–35.

[511] Rohitash Chandra, Abhishek Gupta, Yew-Soon Ong, and Chi-Keong Goh. «Evolutionary multi-task learning for modular training of feedforward neural networks.» In: *Conference on Neural Information Processing.* 2016, pp. 37–46.

[512] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. «Actor-mimic: Deep multitask and transfer reinforcement learning.» In: *arXiv:1511.06342* (2015).

[513] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. «Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.» In: *arXiv:1801.01290* (2018).

[514] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. «Distral: Robust multitask reinforcement learning.» In: *Advances in Neural Information Processing Systems.* 2017, pp. 4496–4506.

[515] Shauharda Khadka and Kagan Tumer. «Evolution-guided policy gradient in reinforcement learning.» In: *Advances in Neural Information Processing Systems*. 2018, pp. 1188–1200.

[516] Joel Lehman and Kenneth O Stanley. «Abandoning objectives: Evolution through the search for novelty alone.» In: *Evolutionary Computation* 19.2 (2011), pp. 189–223.

[517] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. «Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning.» In: *Conference on Robot Learning*. 2020, pp. 1094–1100.

[518] Jacob Andreas, Dan Klein, and Sergey Levine. «Modular multitask reinforcement learning with policy sketches.» In: *International Conference on Machine Learning*. 2017, pp. 166–175.

[519] Yaqing Hou, Yew-Soon Ong, Liang Feng, and Jacek M Zurada. «An evolutionary transfer reinforcement learning framework for multiagent systems.» In: *IEEE Transactions on Evolutionary Computation* 21.4 (2017), pp. 601–615.

[520] Alexander Braylan, Mark Hollenbeck, Elliot Meyerson, and Risto Miikkulainen. «Reuse of neural modules for general video game playing.» In: *AAAI Conference on Artificial Intelligence*. Vol. 30. 2016.

[521] Stephen Kelly and Malcolm I Heywood. «Multi-task learning in atari video games with emergent tangled program graphs.» In: *Genetic and Evolutionary Computation Conference*. 2017, pp. 195–202.

[522] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. «Pathnet: Evolution channels gradient descent in super neural networks.» In: *arXiv:1701.08734* (2017).

[523] AbdElRahman ElSaid, Joshua Karnas, Zimeng Lyu, Daniel Krutz, Alexander G Ororbia, and Travis Desell. «Neuro-Evolutionary Transfer Learning Through Structural Adaptation.» In: *International Conference on the Applications of Evolutionary Computation*. Springer. 2020, pp. 610–625.

[524] Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. «Measuring and Harnessing Transference in Multi-Task Learning.» In: *arXiv:2010.15413* (2020).

[525] Jiabin Lin, Hai-Lin Liu, Bing Xue, Mengjie Zhang, and Fangqing Gu. «Multi-objective Multi-tasking Optimization Based on Incremental Learning.» In: *IEEE Transactions on Evolutionary Computation* 24.5 (2020).

[526] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. «A survey on deep transfer learning.» In: *International Conference on Artificial Neural Networks*. 2018, pp. 270–279.

[527] Matthew E Taylor and Peter Stone. «Transfer learning for reinforcement learning domains: A survey.» In: *Journal of Machine Learning Research* 10.Jul (2009), pp. 1633–1685.

[528] Ling Shao, Fan Zhu, and Xuelong Li. «Transfer learning for visual categorization: A survey.» In: *IEEE Transactions on Neural Networks and Learning Systems* 26.5 (2014), pp. 1019–1034.

[529] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. «To transfer or not to transfer.» In: *NIPS workshop on transfer learning*. Vol. 898. 2005, pp. 1–4.

[530] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. «Characterizing and avoiding negative transfer.» In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11293–11302.

[531] Ruben Glatt, Felipe Leno Da Silva, and Anna Helena Reali Costa. «Towards knowledge transfer in deep reinforcement learning.» In: *Brazilian Conference on Intelligent Systems*. 2016, pp. 91–96.

[532] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. «Emergent tool use from multi-agent autocurricula.» In: *arXiv:1909.07528* (2019).

[533] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. «How transferable are features in deep neural networks?» In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[534] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. «Proximal policy optimization algorithms.» In: *arXiv:1707.06347* (2017).

[535] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. «Trust region policy optimization.» In: *International conference on machine learning*. 2015, pp. 1889–1897.

[536] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. «Learning an embedding space for transferable robot skills.» In: *International Conference on Learning Representations*. 2018.

[537] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. «Multi-Task Reinforcement Learning with Soft Modularization.» In: *arXiv:2003.13661* (2020).

[538] Emanuel Todorov, Tom Erez, and Yuval Tassa. «Mujoco: A physics engine for model-based control.» In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033.

[539] Kevin Frans and Olaf Witkowski. «Population-Based Evolution Optimizes a Meta-Learning Objective.» In: *arXiv preprint arXiv:2103.06435* (2021).

[540] Nadav Kashtan, Elad Noor, and Uri Alon. «Varying environments can speed up evolution.» In: *National Academy of Sciences* (2007).

[541] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. «Generalizing from a few examples: A survey on few-shot learning.» In: *ACM Computing Surveys* (2020).

[542] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. «Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly.» In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[543] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. «Meta-learning in neural networks: A survey.» In: *arXiv preprint arXiv:2004.05439* (2020).

[544] Haoxiang Wang, Han Zhao, and Bo Li. «Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation.» In: *International Conference on Machine Learning.* 2021.

[545] Richa Upadhyay, Prakash Chandra Chhipa, Ronald Phlypo, Rajkumar Saini, and Marcus Liwicki. «Multi-Task Meta Learning: learn how to adapt to unseen tasks.» In: *arXiv preprint arXiv:2210.06989* (2022).

[546] Chelsea Finn, Pieter Abbeel, and Sergey Levine. «Model-agnostic meta-learning for fast adaptation of deep networks.» In: *International conference on machine learning.* 2017.

[547] Richa Upadhyay, Ronald Phlypo, Rajkumar Saini, and Marcus Liwicki. «Sharing to learn and learning to share-Fitting together Meta-Learning, Multi-Task Learning, and Transfer Learning: A meta review.» In: *arXiv preprint arXiv:2111.12146* (2021).

[548] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. «Rl^2: Fast reinforcement learning via slow reinforcement learning.» In: *arXiv preprint arXiv:1611.02779* (2016).

[549] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. «Efficient off-policy meta-reinforcement learning via probabilistic context variables.» In: *International conference on machine learning.* 2019.

[550] Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, OpenAI Jonathan Ho, and Pieter Abbeel. «Evolved policy gradients.» In: *Advances in Neural Information Processing Systems* 31 (2018).

[551] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. «Metapruning: Meta learning for automatic neural network channel pruning.» In: *IEEE/CVF international conference on computer vision.* 2019, pp. 3296–3305.

[552] Juan A Botía, Antonio F Gómez-Skarmeta, Mercedes Valdés, and Antonio Padilla. «Metala: A meta-learning architecture.» In: *International Conference on Computational Intelligence*. Springer. 2001, pp. 688–698.

[553] Pavel Kordík, Jan Černỳ, and Tomáš Frỳda. «Discovering predictive ensembles for transfer learning and meta-learning.» In: *Machine Learning* 107.1 (2018), pp. 177–207.

[554] Alexander Gajewski, Jeff Clune, Kenneth O Stanley, and Joel Lehman. «Evolvability ES: Scalable Evolutionary Meta-Learning.» In: ().

[555] Xingyou Song, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn, and Jie Tan. «Rapidly adaptable legged robots via evolutionary meta-learning.» In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2020, pp. 3769–3776.

[556] Djordje Grbic and Sebastian Risi. «Towards continual reinforcement learning through evolutionary meta-learning.» In: *Genetic and Evolutionary Computation Conference Companion*. 2019, pp. 119–120.

[557] Liwei Wang, Lunjia Hu, Jiayuan Gu, Zhiqiang Hu, Yue Wu, Kun He, and John Hopcroft. «Towards understanding learning representations: To what extent do different neural networks learn the same representation.» In: *Advances in neural information processing systems* 31 (2018).

[558] Saeed Masoudnia and Reza Ebrahimpour. «Mixture of experts: a literature survey.» In: *The Artificial Intelligence Review* 42.2 (2014), p. 275.

[559] Esteban Real, Chen Liang, David R So, and Quoc V Le. «AutoML-Zero: Evolving Machine Learning Algorithms From Scratch.» In: *arXiv preprint arXiv:2003.03384* (2020).

[560] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. «Continual lifelong learning with neural networks: A review.» In: *Neural Networks* 113 (2019), pp. 54–71.

[561] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. «Quality diversity: A new frontier for evolutionary computation.» In: *Frontiers in Robotics and AI* (2016), p. 40.

[562] Jonathan Frankle and Michael Carbin. «The lottery ticket hypothesis: Finding sparse, trainable neural networks.» In: *arXiv preprint arXiv:1803.03635* (2018).

[563] Javier Poyatos, Daniel Molina, Aritz D Martinez, Javier Del Ser, and Francisco Herrera. «EvoPruneDeepTL: An evolutionary pruning model for transfer learning based deep neural networks.» In: *Neural Networks* 158 (2023), pp. 59–82.

[564] Santiago Gonzalez and Risto Miikkulainen. «Effective regularization through loss-function metalearning.» In: *arXiv preprint arXiv:2010.00788* (2020).

[565] Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. «Quality-Diversity Optimization: a novel branch of stochastic optimization.» In: *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*. 2021, pp. 109–135.