Adaptive Multi-factorial Evolutionary Optimization for Multi-task Reinforcement Learning

Aritz D. Martinez, Javier Del Ser, *Senior Member, IEEE,* Eneko Osaba and Francisco Herrera, *Senior Member, IEEE*

***Abstract*—Evolutionary computation has largely exhibited its potential to complement conventional learning algorithms in a variety of Machine Learning tasks, especially those related to unsupervised (clustering) and supervised learning. It has not been until lately when the computational efficiency of evolutionary solvers has been put in prospective for training reinforcement learning models. However, most studies framed so far within this context have considered environments and tasks conceived in isolation, without any exchange of knowledge among related tasks. In this manuscript we present A-MFEA-RL, an adaptive version of the well-known MFEA algorithm whose search and inheritance operators are tailored for multitask reinforcement learning environments. Specifically, our approach includes crossover and inheritance mechanisms for refining the exchange of genetic material, which rely on the multi-layered structure of modern Deep Learning based reinforcement learning models. In order to assess the performance of the proposed approach, we design an extensive experimental setup comprising multiple reinforcement learning environments of varying levels of complexity, over which the performance of A-MFEA-RL is compared to that furnished by alternative non-evolutionary multitask reinforcement learning approaches. As concluded from the discussion of the obtained results, A-MFEA-RL not only achieves competitive success rates over the simultaneously addressed tasks, but also fosters the exchange of knowledge among tasks that could be intuitively expected to keep a degree of synergistic relationship.**

***Index Terms*—Multifactorial Optimization, Evolutionary Multitasking, Multitask Reinforcement Learning, Neuroevolution.**

I. Introduction

Historically, Evolutionary Deep Learning has shown to be a good substitute to traditional Reinforcement Learning training techniques [1], [2]. A usual way to tackle the evolution of neural networks' structure and/or parameters is Neuroevolution (NE) [3], [4]. This research area takes advantage of Evolutionary Algorithms to work towards evolving diverse and well-performing neural networks. Although it has been applied to Reinforcement Learning and video-game playing [5], [6], [7], [8], NE has also been used in other fields such as image classification [9], [10], [11], unveiling the potential of this kind of evolutionary approaches in the current Machine Learning context [12]. In fact, the spectrum of Evolutionary algorithms applied to Deep Learning is prominently expanding [13], [14], [15], [16], with knowledge transfer and multitask learning among the main challenges currently under the focus of Evolutionary Deep Learning [17].

Reinforcement Learning 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*. In its seminal form, Transfer Learning 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 manifold learning problems.

Among such problems, we focus on multitask Reinforcement Learning, in which the objective is to train a model or a set of models that can generalize to multiple reinforcement tasks [18]. Hence, the design goal is to quickly adapt the model to previously unseen problems instances by taking advantage of the knowledge captured during the training phase. For instance, in multi-headed approaches [19] this information can be exchanged in the form of shared neural layers, or by means of distilled policies [20] as in multi-model approaches. Other strategies for promoting exploration in Reinforcement Learning consist of the hybridization of the model with Evolutionary Computation, yielding the concept of Evolutionary Reinforcement Learning. In this line of work, data diversification is generated in [21] 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 [15], 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. Evolutionary Algorithms also provide many opportunities to promote the diversity of solutions. An example can be found in [22], where Reinforcement Learning maze navigation task is evolved via Differential Evolution while taking advantage of the diversity generated by the novelty search mechanism to avoid stagnation. Similarly, novelty search has been applied in [23] to increase diversity on evolving different types of walkers.

In our above elaborations we have reviewed how Transfer Learning can contribute to the effective transfer of knowledge

A. D. Martinez, J. Del Ser and E. Osaba are with TECNALIA, Basque Research & Technology Alliance (BRTA), 48160 Bizkaia, Spain. E-mail: {aritz.martinez,javier.delser,eneko.osaba}@tecnalia.com.

J. Del Ser is also with the University of the Basque Country (UPV/EHU).

F. Herrera is with the Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, 18071 Granada, Spain. E-mail: herrera@decsai.ugr.es.

Manuscript received December 10th, 2020; revised March 25th, 2021.

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 so called Transfer Optimization [24]. Within this paradigm, this paper falls beneath the scope of *multitasking optimization*, which gravitates on solving multiple problems simultaneously by exploiting their underlying synergistic relationships. Further down this literature strand, *Evolutionary Multitasking* resort to concepts and operators from Evolutionary Computation to implement multitasking optimization, with Multifactorial Optimization (MFO) as the most echoing formulation of the multitasking optimization problem considered by the research community working in Evolutionary Multitasking. MFO was first introduced in [25] to formulate the transfer of knowledge between a set of simultaneously solved optimization problems. To realize this strategy, Multifactorial Optimization Algorithm (MFEA) was proposed, garnering since then a high interest due to its capability to effectively leverage the relationships between the problems to be optimized. Years thereafter, an improved variant of this approach (MFEA-II [26]) was proposed to adapt the parameters guiding the knowledge transfer among tasks, thereby reducing negative interactions between unrelated problems.

The work presented in this manuscript adapts the search mechanisms of MFEA to multitask Reinforcement Learning scenarios, including the adaptation of the parameters dictating the exchanged knowledge among tasks during the search. Specifically, we design and implement an evolutionary approach for multitasking reinforcement learning (A-MFEA-RL), which 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 Transfer Learning between several reinforcement learning 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. The novel aspects that differentiate A-MFEA-RL from previous approaches towards achieving the objectives portrayed in this section can be summarized as follows:

- *Design of the unified space towards favoring model-based Transfer Learning*: specifically, aspects such as the neural network architecture, the number of neurons of each layer, and the presence of shared layers among models evolved for each task are taken into account.
- *Adapted crossover operator*: the crossover operator must support the previous aspects by preventing neural models from exchanging irrelevant information.
- *Layer-wise Transfer Learning*: unlike in traditional means to implement Transfer Learning, the number of layers to be transferred between models evolved for different tasks is autonomously decided by A-MFEA-RL during the search.

In order to assess the performance of our proposed approach, we conduct an extensive experimentation seeking to answer three crucial *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 [27], 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 reinforcement learning is conclusive: our approach not only gets competitive results with respect to such *avant-garde* modeling counterparts, but also excels at effectively transferring knowledge among synergistically related reinforcement learning tasks.

This manuscript extends our preliminary findings reported in [28] in several directions, yielding substantially improved aspects with respect to this previous work:

- A completely different experimentation where the reinforcement learning environments are upgraded to much more complex tasks. Also, the number of simultaneously evolved tasks increases from 9 to 50 in the largest setup.
- In order to address more complex tasks, the evolved models meet the scales of realistic deep neural architectures, comprising thousands of parameters.
- An improved design of the unified search space suitable for the case when the shared layers have a significantly different amount of neurons, as well as a crossover operator suited to deal with this redefined search space.
- An adaptation mechanism that learns the intensity of knowledge transferred among tasks in a per layer basis, which is partly inspired by the seminal fashion in which Transfer Learning is realized in neural computation.

The rest of the manuscript is organized as follows. In Section II we overview the state of the art on evolutionary reinforcement learning and multitask learning, whereas Section III establishes fundamental definitions and concepts of reinforcement learning and MFO. Section IV describes in detail the proposed A-MFEA-RL approach, its search space and the rationale for the design of its operators. Next, Section V presents the experimental setup, poses the research questions to be answered, and discusses the obtained results. Finally, concluding remarks are given in Section VI.

II. Related Work

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

There are multiple ways to share knowledge in non-evolutionary multitask reinforcement learning. Multitask multi-headed networks [27], [19] 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 [29] 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 [20] 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 reinforcement learning models.

Shifting the focus on evolutionary knowledge transfer, many approaches conduct genetic knowledge transfer based on different strategies. In [30] and [31] 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 [31], 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 [32] 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 [33] 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 [34], 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 reinforcement learning tasks. A similar approach to the one proposed in this manuscript is presented in [35], where knowledge is transferred among networks and the inputs and outputs are adapted while the internal structure is evolved via neuroevolution. The recurrent networks' structure is evolved by an adaptation of the EXAMM neuroevolution 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 Evolutionary Multitasking 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 [36], 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.

We now proceed by revising works where MFEA is used for evolving problems related to Evolutionary Multitask Reinforcement Learning. In [37], the intrinsic knowledge transferred by MFEA is used to efficiently discover the best paths for multi-UAV path planning tasks. Similarly, in [38] 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 [14], [39] 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 reinforcement learning, memetic evolution, hybrid approaches, multifactorial optimization or neuroevolution), 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 [40], an attempt at gauging the shared inter-task knowledge in multitask reinforcement learning 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 reinforcement learning.

III. Preliminaries

Before proceeding further, this section defines essential concepts in evolutionary Deep Reinforcement Learning, MFO and Transfer Learning, which are of utmost necessity for properly understanding the proposed A-MFEA-RL algorithm.

A. Evolutionary Deep Reinforcement Learning

In its most general formulation, optimization variables in evolutionary Deep Learning represent the hyper-parameters of the model [41], [42] (γ), its neural architecture [43], [44] (number of neurons of each layer ϕ, set of possible connection pattern ω) and/or the set θ of trainable parameters [45], [46] (i.e. weights and biases). The values of such variables are optimized based on a fitness function $f(\cdot)$, which often relate to performance metrics of the network, such as loss or accuracy over a certain holdout dataset. If the fitness represents a measure of performance that must be minimized (e.g. error, loss), the problem can be formulated as:

$$\min_{\mathbf{p}\in\mathcal{P}} f(M_{\mathbf{p}}(\mathcal{D})), \tag{1}$$

where $\mathcal{P} \subseteq \{\gamma, \phi, \omega, \theta\}$ denotes the set of possible variables to be optimized, $\mathbf{p}$ an element of $\mathcal{P}$, and D is the dataset in which the model $M_{\mathbf{p}}(\cdot)$ is trained as per the set of optimization parameters $\mathbf{p}$. Bearing these definitions in mind, Evolutionary

Deep Learning refers to the adoption of Evolutionary Algorithms and Swarm Intelligence methods to tackle the above optimization problem efficiently [17].

This work considers specifically the optimization of the trainable parameters $\boldsymbol{\theta}$ of deep neural networks used for reinforcement learning. 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 $\texttt{env}$. 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 reinforcement learning 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 [2].

Going back to Expression (1), the problem in evolutionary Deep Reinforcement Learning can be reformulated as:

$$\max_{\boldsymbol{\theta}\in\theta} f_r(\texttt{state}(\texttt{env}, \texttt{S}; \pi_{\boldsymbol{\theta}})), \tag{2}$$

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 $\texttt{env}$ for a certain amount of steps $\texttt{S}$. 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 $\texttt{env}$. It is important to note that the reward function is often determined by the environment and the task under consideration.

A glimpse at the literature about evolutionary Reinforcement Learning evinces that many contributions have resorted to neuroevolution as the optimization algorithm addressing the problem in (2). Such works used evolutionary algorithms mainly to jointly train and evolve small network architectures (agents) for low-complexity tasks like Cartpole [4], [13] to more complicated environments [47]. However, the trend in this field is progressively drifting towards collaborative approaches [15] and Transfer Learning, both at architectural [35] and parametric levels [28]. It is in this new playground where MFO can constitute a novel and promising alternative.

B. Multifactorial Evolutionary Algorithm

As stated in the introduction, MFEA [25] 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. The unified space also eases the process of sharing genetic information between tasks. By virtue of its set of evolutionary operators, MFEA finds a set of solutions that solve each of the individual optimization problems defined in $\mathcal{T}$. In order to be able to rank and select the best performing candidates for each task, a population of P individuals $\{\mathbf{x}^p\}_{p=1}^P$ is evolved over the unified search space $\boldsymbol{\mathcal{X}}^U$ based on several key definitions:

- *Skill factor* $\tau^p \in \{1, \ldots, K\}$, defined as the index of the task to which the candidate $\mathbf{x}^p$ is associated. In MFEA candidates are exclusively evaluated on this task for the sake of computational efficiency.
- *Factorial cost* $\Psi_k^p \in \mathbb{R}$ which, for a given task, denotes the fitness value of candidate $\mathbf{x}^p$ decoded and evaluated for task T_k, i.e. $\Psi_k^p = F_k(\mathbf{x}_k^p)$, where $F_k : \boldsymbol{\mathcal{X}}_k \mapsto \mathbb{R}$ is the objective function of task T_k, and $\mathbf{x}_k^p$ the decoded individual of $\mathbf{x}^p$.
- *Factorial rank* $r_k^p \in \{1, \ldots, P\}$, which is simply the relative position of the candidate $\mathbf{x}^p$ in the list of population individuals sorted in ascending order with respect to Ψ_k^p.
- *Scalar fitness* $\varphi^p \in \mathbb{R}[0, 1]$, which is a value assigned to every individual in the population computed based on the task in which they are best ranked, i.e. $\varphi^p = 1/\left(\min_k r_k^p\right)$.

Although the above definitions 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 [26]).

Since its inception, MFEA and MFEA-II have gained increasing popularity within the community [48], [49], [50]. Nonetheless, they have been scarcely used for simultaneously training neural networks [14]. 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 prepared 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.

C. Transfer Learning

Transfer Learning 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 Transfer learning between neural networks under diverse strategies, namely, *instance-based*, *mapping-based*, *adversarial-based* and *network-based* [51]. Network-based approaches have been widely applied in image recognition, single-task reinforcement learning [52], [53] and multitask reinforcement learning [18], [20]. 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* [54], [55].

Even if Transfer Learning 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, Transfer Learning can be very useful for reinforcement learning [56], allowing agents to adapt quickly to new environments. When dealing with extremely complex behavioral policies, agents often require gradual reinforcement learning, by which the agent is sequentially trained to learn progressively more complex tasks. Past knowledge (policies) is reused by means of Transfer Learning towards achieving good policies for the most convoluted task faster [57]. 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 Transfer Learning mechanism.

IV. Proposed approach: A-MFEA-RL

In this section we delve into A-MFEA-RL, the proposed evolutionary multitasking approach capable of evolving multiple reinforcement learning 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 our approach 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 IV-A), the adaptation of the crossover operator (Section IV-B) and how adaptive Transfer Learning is updated during the multitasking search process (Section IV-C). A reference pseudocode of A-MFEA-RL is given in Algorithm 1.

A. Design of the unified search space

As has been mentioned in Section III-B, knowledge transfer between simultaneously evolved reinforcement learning 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 reinforcement learning 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 reinforcement learning 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 $\mathbf{x}^p \in \mathcal{X}^U$ is divided in two parts:

- A set of φ_{sh} layers that are common to all models, defined as $\mathcal{L}^{sh} = \{L^{sh,1}, \ldots, L^{sh,\varphi_{sh}}\}$, 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 $\mathcal{L}^{sp} = \{\mathcal{L}^{sp}_k\}^K_{k=1}$, which is composed by the concatenation of the layers $\mathcal{L}^{sp}_k = \{L^{sp,1}_k, \ldots, L^{sp,\varphi_{sp,k}}_k\}^K_{k=1}$ 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:

$$|\mathbf{x}^p| = \sum_{\ell=1}^{\varphi_{sh}} |L^{sh,\ell}| + \sum_{k=1}^{K} \sum_{\ell'=1}^{\varphi_{sp,k}} |L^{sp,\ell'}_k|, \tag{3}$$

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, we 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, we have to carefully design the location of task-specific parameters inside the encoded individual $\mathbf{x}^p$ so that weights from a model and biases from another do not overwrite each other when Transfer Learning is performed. Otherwise, the transfer will be incoherent and counterproductive for the search [58].

To circumvent this issue, we define a clear distinction between weights $\mathbf{w}^\ell_k$ and biases $\mathbf{b}^\ell_k$ of layer L^ℓ_k, so that the transfer of knowledge considers the potentially different network architectures among tasks. To mathematically define this tailored decoding process, we define two mapping functions that 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 $\lambda^\ell_{k,\mathbf{b}}(\mathbf{x}^p) = \mathbf{b}^\ell_k$. 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 that for some layers

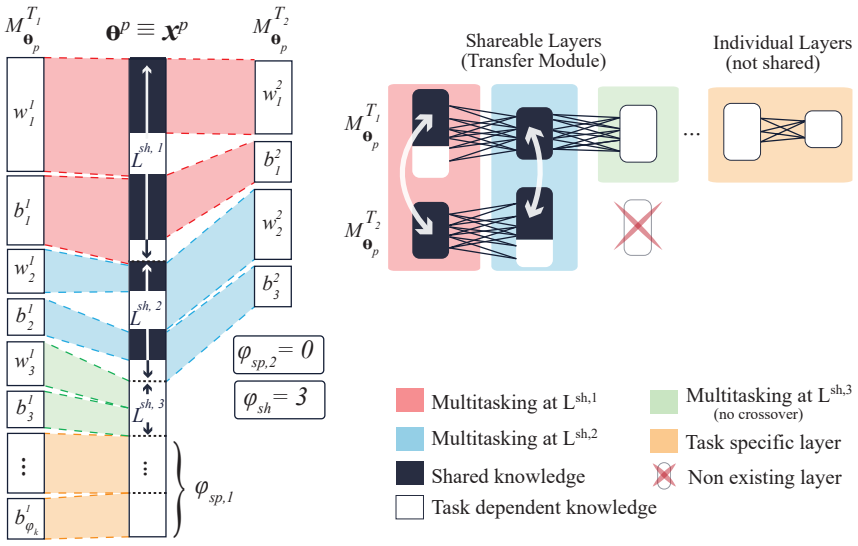

Fig. 1. Schematic diagram illustrating: (left) how the unified search space $\boldsymbol{\mathcal{X}}^U$ is structured in a layer-wise manner, including how it is decoded for two different tasks T_1 and T_2; and (right) how those decoded individuals yield two reinforcement learning models with partly shared weights and biases.

in $\mathbf{x}^p$, $|\mathbf{w}_k^\ell| = |\mathbf{b}_k^\ell| = 0$, 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 $L_{k'}^{sp,\ell}$ with $k' \neq k$. Summarizing, the mapping functions $\lambda_{k,\mathbf{w}}^\ell(\cdot)$ and $\lambda_{k,\mathbf{b}}^\ell(\cdot)$ allow translating every candidate $\mathbf{x}^p$ to the specific network architecture for every task.

Following the schematic example shown in Figure 1 for $K = 2$ tasks, first we observe 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 $\mathbf{w}_k^\ell$ and biases $\mathbf{b}_k^\ell$ 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 Transfer Learning, 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 our 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 Transfer Learning.

B. 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 $\mathcal{L}^{sh}$ of both $\mathbf{x}^p$ and $\mathbf{x}^{p'}$. However, not all shared layers inside $\mathcal{L}^{sh}$ are selected for crossover. Instead, we build on the intuition that 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 [26]. 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 $K \times K \times \mathcal{L}^{sh}$ 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 $rmp_{k,k'}^\ell$, 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 Transfer Learning, 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 $rmp_{k,k}^\ell = 1.0 \ \forall k, \ell$. 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, $rmp_{k,k'}^\ell$ and $rmp_{k',k}^\ell$ 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-

Algorithm 1: Proposed A-MFEA-RL

$\mathcal{T} \leftarrow$ set of tasks $\{T_1, T_2, \ldots, T_K\}$ and $K = |\mathcal{T}|$
Define number of individuals per task as P_k
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 IV-A)***
Generate a population $\mathcal{P}$ of $P = \sum_{k=1}^{K} P_k$ candidates $\{\mathbf{x}^p\}_{p=1}^{P}$, with $\mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$
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 $\{P_k/P\}_{k=1}^{K}$
Set $rmp^{\ell}_{k,k'} = rmp_{ini}$ and $rmp^{\ell}_{k,k} = 1\ \forall \ell$ and $k \neq k'$
Evaluate every $\mathbf{x}^p$ over task T_{τ^p} ***(Section IV-D)***
Set gen $= 1$
while *gen* $\leq$ *maxGen* **do**
  Couple all individuals $\{\mathbf{x}^p\}_{p=1}^{P}$ in pairs at random
  Offspring population $\mathcal{Q} \leftarrow \varnothing$
  Set $C^{\ell,+}_{k,k'} = C^{\ell,-}_{k,k'} = 0$
  for *each coupled pair of individuals* $\mathbf{x}^p$, $\mathbf{x}^{p'}$ **do**
    Select a subset of the shared layers $\mathcal{L}^{sh}$ for crossover as per $rmp^{\ell}_{\tau^p,\tau^{p'}}$
    if *at least one layer to be exchanged* **then**
      $\mathbf{x}^p_o, \mathbf{x}^{p'}_o \leftarrow \text{SBX}(\mathbf{x}^p, \mathbf{x}^{p'})$ ***(Section IV-B)***
      $\tau^p_o \leftarrow \text{randomChoice}(\tau^p, \tau^{p'})$
      $\tau^{p'}_o \leftarrow \text{randomChoice}(\tau^p, \tau^{p'})$
    else
      $\mathbf{x}^p_o \leftarrow \text{polMut}(\mathbf{x}^p)$, $\mathbf{x}^{p'}_o \leftarrow \text{polMut}(\mathbf{x}^{p'})$
      $\tau^p_o \leftarrow \tau^p$, $\tau^{p'}_o \leftarrow \tau^{p'}$
    end
    Evaluate $\mathbf{x}^p_o$ on $T_{\tau^p_o}$ ***(Section IV-D)***
    Evaluate $\mathbf{x}^{p'}_o$ on $T_{\tau^{p'}_o}$
    Update $C^{\ell,+}_{k,k'}$, $C^{\ell,-}_{k,k'}$ for $k = \tau^p$ and $k' = \tau^{p'}$
    Store offspring: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathbf{x}^p_o, \mathbf{x}^{p'}_o\}$
  end
  Create combined population $\mathcal{P}' \leftarrow \mathcal{P} \cup \mathcal{Q}$
  Build the new population $\mathcal{P} \subset \mathcal{P}'$ by retaining the best individuals as per their scalar fitness φ^p, and maintaining the task distribution ($\{P_k/P\}_{k=1}^{K}$)
  Update $rmp^{\ell}_{k,k'}$ based on $C^{\ell,+}_{k,k'}$ and $C^{\ell,-}_{k,k'}$ $\forall k, k', \ell$ ***(Section IV-C)***, and set gen $=$ gen $+ 1$
end
Return the best individual in $\mathcal{P}$ for each task T_k

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 mated 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 Transfer Learning 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 we next detail.

C. Adaptive knowledge transfer update

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 Transfer Learning 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).

We now proceed by explaining how $rmp^{\ell}_{k,k'}$ values are updated. First of all, 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:

$$m^{\ell} = \begin{cases} 1 & \text{if } rand < rmp^{\ell}_{k,k'}, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

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:

$$rmp^{l}_{k,k} \leftarrow \texttt{clip}^{rmp_{max}}_{rmp_{min}}\Big(rmp^{l}_{k,k} - \alpha \cdot C^{\ell,-}_{k,k'} + \beta \cdot C^{\ell,+}_{k,k'}\Big) \tag{5}$$

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, we impose lower (rmp_{min}) and upper (rmp_{max}) bounds 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 to fall into a state where a task does not share/import any knowledge.

D. 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. As was shown in Section III-A, 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 reinforcement learning 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 $\mathtt{S}$ 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 E different episodes, so that the fitness value $F_k(\mathbf{x}_k^p)$ of the individual $\mathbf{x}_k^p$ decoded from $\mathbf{x}^p \in \boldsymbol{\mathcal{X}}^U$ for task T_k is computed as the median value of the E reward values obtained after $\mathtt{S}$ steps simulated for each episode.

V. Experimental Setup

As a result of its tailored design, A-MFEA-RL addresses the goal of simultaneously evolving multiple reinforcement learning 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 three *research questions* (RQ):

- RQ1: *Is A-MFEA-RL able to efficiently exploit positive inter-task synergies, and elude negative knowledge transfer?*
- RQ2: *Does A-MFEA-RL perform competitively compared to existing multitask reinforcement learning approaches?*
- RQ3: *Does A-MFEA-RL scale up nicely when more reinforcement learning 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 may serve as a way to validate the performance of the algorithm and the concepts introduced throughout the paper, RQ2 and RQ3 are devoted to the comparison of A-MFEA-RL to other state-of-the-art multitask reinforcement learning methods:

- *Multitask Proximal Policy Optimization* (M-PPO) [59].
- *Multitask Trust Region Policy Optimization* (M-TRPO) [60].
- *Task Embeddings* (TE) [61].
- *Multitask Soft Actor Critic* (M-SAC) [19].
- *Multitask Multi-head Soft Actor Critic* (M-M-SAC) [19].
- *Shallow Multitask Reinforcement Learning with Soft Modularization* (S-MRL-SM) [62].
- *Deep Multitask Reinforcement Learning with Soft Modularization* (D-MRL-SM) [62].

The experimentation is built upon the *Metaworld* framework [27], which implements a set of 50 robot arm based tasks (Figure 2) modeled and run over the MuJoCo physics simulator [63]. 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 [27]. These *Metaworld* environments allow evaluating the performance of developed reinforcement learning 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.

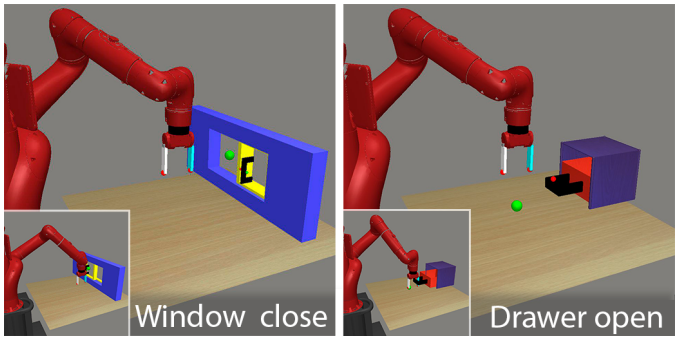

Fig. 2. Images illustrating the `window close` and `drawer open` tasks of the *Metaworld* framework. The ending state of a successfully completed episode is shown in the nested image on the bottom left corner of every plot.

Among the totality of 50 environments embedded in *Metaworld*, some of them are relatively more complicated than others for a reinforcement learning agent to solve them successfully. Accordingly, we have divided the 50 environments 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 I, along with their average number of trainable parameters (weights and biases).

TABLE I
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 II. 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 $rmp_{k,k'}^{\ell}$ 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 reinforcement learning methods considered in RQ2, RQ3 and RQ5 are equal to those utilized in the publications where they were first presented. We run the experimentation 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.

TABLE II
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

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

RQ1: Evaluating the knowledge transfer between tasks

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, we design a simple scenario 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 will hereafter denote such tasks as *twin tasks*.

We start our analysis with Figure 3, where 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 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, if we pause at `window-close`, this 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.

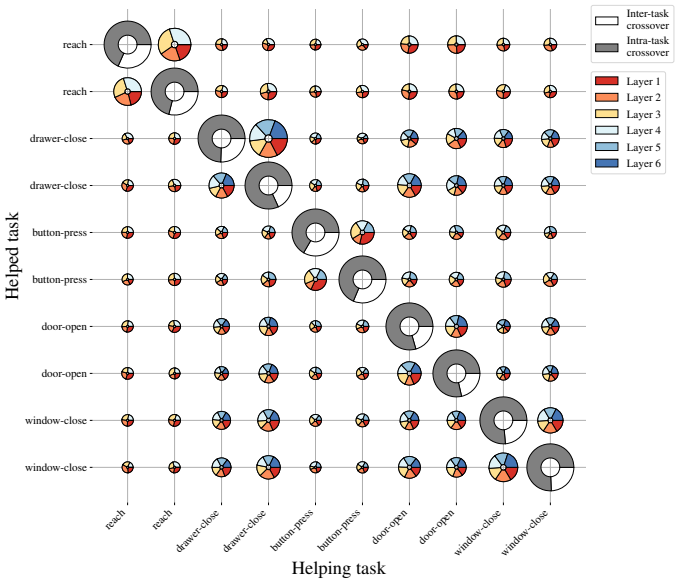

Fig. 3. Bubble plot matrix showing the amount of knowledge transferred between tasks involved in RQ1, for every layer of the neural networks that embody the evolved agents.

When focusing on how the above knowledge exchange is distributed over layers of the networks involved in the transfer, we 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 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

reinforcement learning tasks considered in RQ1. To this end, Table III 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 III
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, we have observed several clear synergies between tasks in this first experimental setup, not only between twin tasks, but also across 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.

RQ2: Comparing A-MFEA-RL to other multitask approaches

We now turn the focus on RQ2, which aims to compare A-MFEA-RL to other state-of-the-art reinforcement learning approaches suited for multitask environments. For this purpose we design a similar multitask scenario as the one used in [62], extending the so-called *MT-10* problem proposed in [27] 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 [27].

Our discussion follows the same flow as the one made in the previous subsection for RQ1. Accordingly, we start by analyzing the *helping* tasks that contributed most for the improvement of individuals specialized in other *helped* tasks. From Figure 4 we can observe 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 [40], 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. 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 [40].

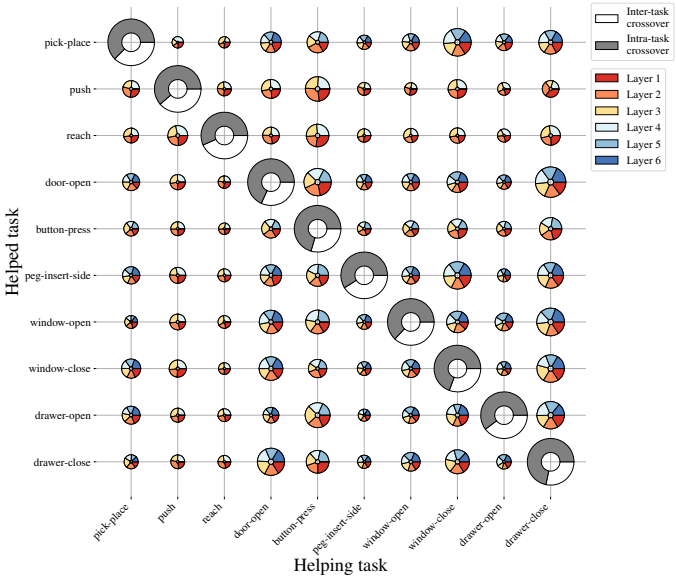

Fig. 4. Bubble plot matrix showing the amount of knowledge transferred between tasks in *MT-10-R*.

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, we now pause at Table IV, which 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. We also add to the comparison 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 I for guaranteeing a fair comparison to *A-MFEA-RL*. These results are complemented by those in Table A.1 included in the appendix, 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 IV
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 [59]	25.0%	-
M-TRPO [60]	29.0%	-
TE [61]	30.0%	-
M-SAC [19]	53.3%	44.4%
M-M-SAC [19]	71.6%	57.7%
S-MRL-SM [62]	73.3%	**74.5%**
D-MRL-SM [62]	73.2%	67.9%
A-MFEA-RL (proposed)	**80.0%**	73.9%

We first focus on single-task SAC, which 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 reinforcement learning tasks harder to be solved, agents evolved for these three environments are expected to perform even worse. Therefore, we leave M-PPO, M-TRPO and TE as the baselines, and center our discussion 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 [19], [62].

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 ($|\mathbf{x}^p| \sim 87$K parameters) than the ones in M-M-SAC (~ 327K), S-MRL-SM and D-MRL-SM (~ 270K).

Summarizing, experiments aimed at answering RQ2 have revealed that A-MFEA-RL elicits a competitive performance when compared to *avant-garde* multitask reinforcement learning 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 between tasks discovered by A-MFEA-RL have been found to remain in close accordance with those recently reported in [40]. In the next section we scale up this experimentation to 50 simultaneously evolved tasks, towards ascertaining whether the competitive performance of A-MFEA-RL holds in more populated multitasking setups.

RQ3: Scaling up the comparison study to more tasks

We now proceed by addressing research question RQ3, which aims to elucidate whether A-MFEA-RL contends with the rest of multitask reinforcement learning methods when the number of simultaneously solved tasks K increases. To this end, we extend the simulation setup with the rest of 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 V
SUCCESS RATE OF TASKS EVOLVED FOR RQ3: MT-50 AND MT-50-R

Algorithm	MT-50	MT-50-R
M-PPO [59]	9.0%	-
M-TRPO [60]	22.9%	-
TE [61]	15.3%	-
M-SAC [19]	26.2%	25.7%
M-M-SAC [19]	28.6%	29.0%
S-MRL-SM [62]	57.3%	61.5%
D-MRL-SM [62]	**62.0%**	**62.1%**
A-MFEA-RL (proposed)	60.0%	59.7%

We concentrate our analysis on the success rates attained by all the methods in the benchmark, which are summarized in Table V. 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*, our approach 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 reinforcement learning models underneath S-MRL-SM and D-MRL-SM.

We round up our discussion by delving into the per-task results obtained for these extended simulation setups, which are listed in Table A.1. 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.

RQ4: Benefits of adaptive knowledge transfer

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, we add to the comparison an adaptation of the traditional MFEA, which has been tuned to utilize the same unified space than A-MFEA-RL. Therefore, the main differences between the adapted MFEA and our proposal 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). We report on 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 [25]. 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 VI
COMPARING A-MFEA-RL TO FIXED RMP VALUES AND MFEA. THE SUCCESS RATES FOR ALL TASKS CAN BE FOUND IN OUR GIT REPOSITORY

	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 ($rmp = 0.3$) [25]	63.1%	26.2%
A-MFEA-RL (adaptive)	**73.9**%	**59.7**%

Table VI summarizes the results of the experiments run for answering RQ4. In this table we can first observe 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, our 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.

RQ5: On the computational cost of A-MFEA-RL

We end up our experiments 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. Our approach is compared to that of *M-SAC* [19], 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 VII 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 $\mathrm{E} = 3$ 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, 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 metaheuristics on GPU processors [64], and stimulates future efforts towards GPU implementations of evolutionary multitask approaches to benefit from the computational efficiency of these massively parallel hardware architectures.

VI. CONCLUSIONS AND RESEARCH DIRECTIONS

This manuscript has elaborated on multitask reinforcement learning, which aims at simultaneously learning to perform multiple tasks by exploiting the potential synergies existing among them. In this context, we have presented A-MFEA-RL,

TABLE VII
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			M-SAC			M-SAC-GPU		
	Single Task	MT10	MT50	Single Task	MT10	MT50	Single Task	MT10	MT50
train_time	3.6e-1±6.4e-2	4.2±7.4e-1	19.5±6.1e-1	34.82±4.27	389.53±7.5	1933.26±12.5	2.9±3.2e-2	42.78±11.76	301±16.28
eval_time	23.53±4.68	216.9±2.26	1036.3±8.7	2e-3±3e-4	5.23e-3 ± 8.9e-4	2e-1±3e-3	1.2e-3±1.3e-4	3.1e-3±4.7e-4	1e-2±7.8e-4

an evolutionary multitasking algorithm that has been adapted to deal with multitask reinforcement learning 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 reinforcement learning 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, our approach 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 reinforcement learning, both in terms of quality of solutions and computational cost.

Several research lines rooted on this work are planned. To begin with, A-MFEA-RL can be extrapolated to other learning paradigms in which Transfer Learning has been shown to be crucial, including generative and discriminative problems (e.g. image classification). Furthermore, we envision that the information contained in the learned rmp tensors can be further exploited to develop multi-parent crossover operators involving multiple tasks in the breeding process. Furthermore, we also plan to study the impact of new tasks introduced in the optimization process in an online fashion. All these research lines embody the core of our research agenda in the future.

ACKNOWLEDGMENTS

A. D. Martinez, J. Del Ser and E. Osaba acknowledge the funding support received from the Basque Government through the ELKARTEK program (3KIA project, KK-2020/00049). J. Del Ser also acknowledges funding support from the Consolidated Research Group MATHMODE (IT1294-19) granted by the Department of Education of the Basque Government. F. Herrera would like to thank the Spanish Government for its funding support (SMART-DaSCI, TIN2017-89517-P), as well as the BBVA Foundation through its Ayudas Fundación BBVA a Equipos de Investigación Científica 2018 call (DeepSCOP).

APPENDIX

A. Individual success rates per task (next page)

REFERENCES

[1] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv:1712.06567*, 2017.

[2] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "Back to basics: Benchmarking canonical evolution strategies for playing atari," *arXiv:1802.08842*, 2018.

[3] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.

[4] K. O. Stanley and R. Miikkulainen, "Efficient reinforcement learning through evolving neural network topologies," in *Annual Conference on Genetic and Evol. Comput.*, pp. 569–577, 2002.

[5] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *IEEE Trans. Evol. Comput.*, vol. 9, no. 6, pp. 653–668, 2005.

[6] J.-B. Mouret and S. Doncieux, "Encouraging behavioral diversity in evolutionary robotics: An empirical study," *Evol. Comput.*, vol. 20, no. 1, pp. 91–133, 2012.

[7] A. P. Poulsen, M. Thorhauge, M. H. Funch, and S. Risi, "Dlne: A hybridization of deep learning and neuroevolution for visual control," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 256–263, IEEE, 2017.

[8] Y. Tang, D. Nguyen, and D. Ha, "Neuroevolution of self-interpretable agents," in *Genetic and Evol. Comput. Conference*, pp. 414–424, 2020.

[9] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*, pp. 2902–2911, 2017.

[10] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4780–4789, 2019.

[11] J. Liang, E. Meyerson, B. Hodjat, D. Fink, K. Mutch, and R. Miikkulainen, "Evolutionary neural automl for deep learning," in *Genetic and Evol. Comput. Conference*, pp. 401–409, 2019.

[12] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.

[13] C. Igel, "Neuroevolution for reinforcement learning using evolution strategies," in *IEEE Congress on Evol. Comput.*, pp. 2588–2595, 2003.

[14] R. Chandra, A. Gupta, Y.-S. Ong, and C.-K. Goh, "Evolutionary multitask learning for modular training of feedforward neural networks," in *Conference on Neural Information Processing*, pp. 37–46, 2016.

[15] S. Khadka, S. Majumdar, T. Nassar, Z. Dwiel, E. Tumer, S. Miret, Y. Liu, and K. Tumer, "Collaborative evolutionary reinforcement learning," *arXiv:1905.00976*, 2019.

[16] J. Koutník, J. Schmidhuber, and F. Gomez, "Evolving deep unsupervised convolutional networks for vision-based reinforcement learning," in *Annual Conference on Genetic and Evol. Comput.*, pp. 541–548, 2014.

[17] A. D. Martinez, J. Del Ser, E. Villar-Rodriguez, E. Osaba, J. Poyatos, S. Tabik, D. Molina, and F. 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.

[18] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv:1511.06342*, 2015.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv:1801.01290*, 2018.

TABLE A.1
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 REPRESENTED AS E (*Easy*), M (*Medium*) AND H (*Hard*)

	MT-10			**MT-10-R**			**MT-50**			**MT-50-R**		
Environment name (complexity)	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

[20] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," in *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

[21] S. Khadka and K. Tumer, "Evolution-guided policy gradient in reinforcement learning," in *Advances in Neural Information Processing Systems*, pp. 1188–1200, 2018.

[22] K. Mason, J. Duggan, and E. Howley, "Maze navigation using neural networks evolved with novelty search and differential evolution," in *Adaptive and Learning Agents Workshop*, 2018.

[23] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evol. Comput.*, vol. 19, no. 2, pp. 189–223, 2011.

[24] A. Gupta, Y.-S. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 1, pp. 51–64, 2017.

[25] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 343–357, 2015.

[26] K. K. Bali, Y.-S. Ong, A. Gupta, and P. S. Tan, "Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 69–83, 2019.

[27] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*, pp. 1094–1100, 2020.

[28] A. D. Martinez, E. Osaba, J. Del Ser, and F. Herrera, "Simultaneously evolving deep reinforcement learning models using multifactorial optimization," in *IEEE Congress on Evol. Comput.*, pp. 1–8, 2020.

[29] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *International Conference on Machine Learning*, pp. 166–175, 2017.

[30] L. Feng, Y.-S. Ong, A.-H. Tan, and I. W. Tsang, "Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems," *Memetic Computing*, vol. 7, no. 3, pp. 159–180, 2015.

[31] Y. Hou, Y.-S. Ong, L. Feng, and J. M. Zurada, "An evolutionary transfer reinforcement learning framework for multiagent systems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 601–615, 2017.

[32] A. Braylan, M. Hollenbeck, E. Meyerson, and R. Miikkulainen, "Reuse of neural modules for general video game playing," in *AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[33] S. Kelly and M. I. Heywood, "Multi-task learning in atari video games with emergent tangled program graphs," in *Genetic and Evol. Comput. Conference*, pp. 195–202, 2017.

[34] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv:1701.08734*, 2017.

[35] A. ElSaid, J. Karnas, Z. Lyu, D. Krutz, A. G. Ororbia, and T. Desell, "Neuro-evolutionary transfer learning through structural adaptation," in *International Conference on the Applications of Evol. Comput. (Part of EvoStar)*, pp. 610–625, Springer, 2020.

[36] Y. Feng, L. Feng, Y. Hou, and K. C. Tan, "Large-scale optimization via evolutionary multitasking assisted random embedding," in *IEEE Congress on Evol. Comput.*, pp. 1–8, 2020.

[37] Y.-S. Ong and A. Gupta, "Evolutionary multitasking: a computer science view of cognitive multitasking," *Cognitive Computation*, vol. 8, no. 2, pp. 125–142, 2016.
[38] Y. Zhou, T. Wang, and X. Peng, "MFEA-IG: A multi-task algorithm for mobile agents path planning," in *IEEE Congress on Evol. Comput.*, pp. 1–7, 2020.
[39] R. Chandra, A. Gupta, Y.-S. Ong, and C.-K. Goh, "Evolutionary multi-task learning for modular knowledge representation in neural networks," *Neural Processing Letters*, vol. 47, no. 3, pp. 993–1009, 2018.
[40] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn, "Measuring and harnessing transference in multi-task learning," *arXiv:2010.15413*, 2020.
[41] G. H. De Rosa, J. P. Papa, and X.-S. Yang, "Handling dropout probability estimation in convolution neural networks using meta-heuristics," *Soft Computing*, vol. 22, no. 18, pp. 6147–6156, 2018.
[42] B. Guo, J. Hu, W. Wu, Q. Peng, and F. Wu, "The tabu_genetic algorithm: a novel method for hyper-parameter optimization of learning algorithms," *Electronics*, vol. 8, no. 5, p. 579, 2019.
[43] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification," in *IEEE Congress on Evol. Comput.*, pp. 1–8, 2018.
[44] Y.-L. Hu and L. Chen, "A nonlinear hybrid wind speed forecasting model using lstm network, hysteretic elm and differential evolution algorithm," *Energy conversion and management*, vol. 173, pp. 123–142, 2018.
[45] S. Alvernaz and J. Togelius, "Autoencoder-augmented neuroevolution for visual doom playing," in *IEEE Conference on Computational Intelligence and Games*, pp. 1–8, 2017.
[46] V. Ayumi, L. R. Rere, M. I. Fanany, and A. M. Arymurthy, "Optimization of convolutional neural network using microcanonical annealing algorithm," in *International Conference on Advanced Computer Science and Information Systems*, pp. 506–511, 2016.
[47] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone, "A neuroevolution approach to general atari game playing," *IEEE Trans. Comput. Intell. AI in Games*, vol. 6, no. 4, pp. 355–366, 2014.
[48] J. Lin, H.-L. Liu, B. Xue, M. Zhang, and F. Gu, "Multi-objective multi-tasking optimization based on incremental learning," *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, 2020.
[49] L. Zhou, L. Feng, K. C. Tan, J. Zhong, Z. Zhu, K. Liu, and C. Chen, "Toward adaptive knowledge transfer in multifactorial evolutionary computation," *IEEE Trans. Cybern.*, 2020.
[50] P. D. Thanh, H. T. T. Binh, and T. B. Trung, "An efficient strategy for using multifactorial optimization to solve the clustered shortest path tree problem," *Applied Intelligence*, vol. 50, no. 4, pp. 1233–1258, 2020.
[51] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, pp. 270–279, 2018.
[52] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1633–1685, 2009.
[53] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019–1034, 2014.
[54] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, "To transfer or not to transfer," in *NIPS workshop on transfer learning*, vol. 898, pp. 1–4, 2005.
[55] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell, "Characterizing and avoiding negative transfer," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11293–11302, 2019.
[56] R. Glatt, F. L. Da Silva, and A. H. R. Costa, "Towards knowledge transfer in deep reinforcement learning," in *Brazilian Conference on Intelligent Systems*, pp. 91–96, 2016.
[57] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," *arXiv:1909.07528*, 2019.
[58] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.
[59] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.
[60] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, 2015.
[61] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *International Conference on Learning Representations*, 2018.
[62] R. Yang, H. Xu, Y. Wu, and X. Wang, "Multi-task reinforcement learning with soft modularization," *arXiv:2003.13661*, 2020.
[63] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.
[64] M. Essaid, L. Idoumghar, J. Lepagnot, and M. Brévilliers, "GPU parallelization strategies for metaheuristics: a survey," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, no. 5, pp. 497–522, 2019.

Aritz D. Martinez studied Telecommunications Engineering at the University of the Basque Country (UPV/EHU, Spain), he obtained his M.Sc. degree in Computational Engineering and Intelligent Systems from this university in 2018 and now he is working towards his PhD at Tecnalia Research and Innovation. His research interests are focused on heuristic techniques for optimization, parallel computing, delay tolerant networking, Deep Learning, Reinforcement Learning and Federated Machine Learning, among others.

Javier Del Ser (SM'12) received his first PhD degree (cum laude) in Electrical Engineering from the University of Navarra (Spain) in 2006, and a second PhD degree (cum laude, extraordinary PhD prize) in Computational Intelligence from the University of Alcala (Spain) in 2013. He is currently a Research Professor in Artificial Intelligence and leading scientist of the OPTIMA (Optimization, Modeling and Analytics) research area at TECNALIA, Spain. He is also an adjunct professor at the University of the Basque Country (UPV/EHU). His research interests are in the design of artificial intelligence methods for data mining and optimization applied to problems in industry, energy, mobility and health, among others. He has published more than 380 scientific articles, co-supervised 10 Ph.D. theses, edited 7 books and co-authored 9 patents. He is an Associate Editor of tier-one journals from areas related to Artificial Intelligence, and a recipient of the Bizkaia Talent prize for his research career.

Eneko Osaba works at TECNALIA as researcher in the ICT/OPTIMA area. He obtained his Ph.D. degree on Artificial Intelligence in 2015 in the University of Deusto. Throughout his career, he has participated in the proposal, development and justification of more than 20 local and european research projects. Additionally, Eneko has also participated in the publication of 120 scientific papers (including more than 25 Q1). Eneko has served as member of the program committee in more than 40 international conferences. Furthermore he has participated in the organization of several international conferences, and he is member of the editorial board of several international journals.

Francisco Herrera (SM'15) received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada and Director of DaSCI Institute (Andalusian Research Institute in Data Science and Computational Intelligence). His current research interests include among others, Computational Intelligence (including fuzzy modeling, computing with words, evolutionary algorithms and deep learning), information fusion and decision making, and data science (including data preprocessing, prediction and big data). He has been the supervisor of more than 50 Ph.D. students. He has published more than 500 journal papers, receiving more than 95,000 citations (Google Scholar, H-index 151). He is co-author of several books, and the Editor in Chief of Information Fusion (Elsevier). He is been selected as a Highly Cited Researcher (in the fields of Computer Science and Engineering, 2014 to present, Clarivate Analytics).