Multitask Neuroevolution for Reinforcement Learning With Long and Short Episodes

Nick Zhang®, Abhishek Gupta®, Senior Member, IEEE, Zefeng Chen®, and Yew-Soon Ong®, Fellow, IEEE

Abstract—Studies have shown evolution strategies (ES) to be a promising approach for reinforcement learning (RL) with deep neural networks. However, the issue of high sample complexity persists in applications of ES to deep RL over long horizons. This article is the first to address the shortcoming of today’s methods via a novel neuroevolutionary multitasking (NuEMT) algorithm, designed to transfer information from a set of auxiliary tasks (of short episode length) to the target (full length) RL task at hand. The auxiliary tasks, extracted from the target, allow an agent to update and quickly evaluate policies on shorter time horizons. The evolved skills are then transferred to guide the longer and harder task toward an optimal policy. We demonstrate that the NuEMT algorithm achieves data-efficient evolutionary RL, reducing expensive agent-environment interaction data requirements. Our key algorithmic contribution in this setting is to introduce a first multitask skills transfer mechanism based on the statistical importance sampling technique. In addition, an adaptive resource allocation strategy is utilized to assign computational resources to auxiliary tasks based on their gleaned usefulness. Experiments on a range of continuous control tasks from the OpenAI Gym confirm that our proposed algorithm is efficient compared to recent ES baselines.

Index Terms—Evolution strategies (ESs), evolutionary multitasking (EMT), reinforcement learning (RL).

Manuscript received 9 June 2022; revised 26 September 2022; accepted 30 October 2022. Date of publication 14 November 2022; date of current version 11 September 2023. This work was supported in part by the Data Science and Artificial Intelligence Research Center (DSAIR), School of Computer Science and Engineering, Nanyang Technological University; in part by A*STAR A13 Seed Grant under Grant C211118016: in part by A*STAR RIE2020 IAF-PP under Grant A19C1a0018; in part by the A*Star Center for Frontier AI Research; and in part by the National Natural Science Foundation of China under Grant 62206313. (Corresponding author: Nick Zhang.)

Nick Zhang is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: nick0007@e.ntu.edu.sg).

Abhishek Gupta is with the Singapore Institute of Manufacturing Technology, Agency for Science, Technology and Research, Singapore 138634, and also with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: abhishek_gupta@simtech.a-star.edu.sg).

Zefeng Chen is with the School of Artificial Intelligence, Sun Yat-sen University, Guangzhou 510275, China, and also with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: chenzef5@mail.sysu.edu.cn).

Yew-Soon Ong is with the Data Science and Artificial Intelligence Research Centre, School of Computer Science and Engineering, Nanyang Technological University, Singapore, and also with the Center for Frontier AI Research, Agency for Science, Technology and Research, Singapore (e-mail: asysong@ntu.edu.sg).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCDS.2022.3221805.

Digital Object Identifier 10.1109/TCDS.2022.3221805

I. INTRODUCTION

IN REINFORCEMENT learning (RL), agents learn how to interact with dynamic and uncertain environments by taking actions that maximize expected rewards. To this end, many popular RL algorithms, such as TRPO [1], DDPG [2], D4PG [3], DQN [4], among others, are based on the Markov decision process (MDP) formalism and the concept of value functions. Algorithms belonging to this class have achieved notable success in applications, such as locomotion tasks [1], [5], strategy board games (e.g., surpassing the level of the world champion in GO [6]) and playing Atari from pixels [1], [4], [5].

Advances in RL have however shed light on evolution strategies (ESs) as a viable alternative for handling continuous control tasks [7], or even playing Atari games with pixel inputs [8]. Being a derivative-free approach, ES attempts to directly search for an optimal policy mapping the states of an environment to the best course of action, with the objective of maximizing the agent’s cumulative rewards. This type of approach falls under the umbrella of neuroevolution when the underlying policy is parameterized by a neural network [9]. The simplicity of implementation (obviating the need to propagate derivatives through the computation graph), better exploration capacity, and inherently parallelizable nature of population-based search makes ES a scalable option for RL, especially given access to distributed computer systems.

Despite the simplicity, surprising levels of performance achievable, and scalability of ES—delivering linear speedup in the number of CPU cores available—the high sample complexity of such black-box optimization methods leaves much to be desired. Even though ES can make good use of parallel resources, the need for vast amounts of agent-environment interaction data to evaluate populations of candidate policies can become prohibitively resource intensive. Moreover, each data sample generated in a standard ES is utilized just for a single search update, hence indicating a wastage of costly and useful information. The issue of lowering sample complexity has thus attracted significant research interest lately, with various methods proposed for making effective use of data in the search for optimal policies [10], [11]. One promising approach in this regard is that of experience transfer [12]. The motivation derives from an analogy to humans who, instead of solving problems from scratch, learn to adapt and reuse experiential priors. Likewise, it is deemed that RL agents may also learn more efficiently by building on data/experiences drawn from related tasks. Algorithmic realizations of this concept...
have even been explored beyond the realms of RL, for teaching machines transferable skills in diverse areas of learning [13] and optimization [14].

Tapping on experience transfers in deep RL, Fuks et al. [15] proposed the idea of progressive episode lengths (PEL) with canonical ES [16]. PEL comprises a set of artificially generated auxiliary tasks, extracted from the target RL environment, that are distinguished by the ascending order of their episode lengths. The tasks are then tackled sequentially, with the search for each task seeded by good solutions found for its predecessor. This procedure naturally leads to the reuse of experiential priors from related (short time horizon) tasks, with the hope of speeding up learning on longer and harder tasks. However, the direct seeding of solutions could cause harmful negative transfers. Evolved skills may overspecialize to an auxiliary task, and hence not generalize well to the target (full length) task at hand. As illustration, imagine a hypothetical agent to be trained for a 26.2-mile marathon, where energy is to be strategically conserved. If such an agent first specializes on shorter 100-m sprints, then the burst in speed would be energy sapping, leaving little in the tank for the full marathon.

In this article, we thus re-examine experience transfers for the first time in the light of evolutionary multitasking (EMT) [17], [18], where an evolving population is uniquely leveraged to tackle multiple tasks (auxiliary and the target) concurrently with periodic exchange of generated solutions. Our aim is to achieve data-efficient evolutionary RL by effectively exploiting transferable skills while guarding against threats of negative transfer. Minimizing wastage of computational resources on auxiliary tasks that are not useful to the target is a key consideration. To this end, we propose a new EMT algorithm that builds on a popular variant of ES (labeled as the OpenAI-ES [8]) with known efficacy for direct neural network policy search. We refer to our neuroevolutionary multitasking NuEMT algorithm as NuEMT for short. The crucial distinction from [15] is that instead of processing tasks sequentially, neuroevolutionary multitasking (NuEMT) combines all auxiliary (short episode length) tasks with the target in a single multitask formulation. This not only supports sample efficiency by learning from relevant experiential priors, but also enables online neutralization of those tasks whose evolved skills do not transfer well to the target agent.

Since the inception of EMT, a range of techniques using probability mixture models (to capture distributional overlaps of jointly evolving populations) have been proposed to adapt the quantity and frequency of intertask information transfers [19], with successful application in [20], [21], and [22]. However, the need to rebuild mixture models at prespecified “transfer intervals” can be expensive, bringing an additional layer of internal algorithmic complexity to the search. To overcome this bottleneck, we equip NuEMT with a novel yet simple multitask transfer mechanism based on importance sampling [23]. Stochastic gradient estimates with cross-task solution sampling and fast mixture model updates are made possible, without the need to continually rebuild the model from the ground up. What’s more, stochastic updates of the mixture coefficients yield the evolving relevance of the auxiliary tasks. Hence, the coefficient values are used to allocate more computational resources to those tasks that are gleaned to be useful to the target, while neutralizing those that are not.

In sum, the main contributions of this article, including the problem setup and the proposed algorithm, are fourfold.

1) An EMT formulation with skills transfer from tasks with shorter to longer episode lengths is presented.
2) A novel NuEMT algorithm with importance sampling for intertask experience transfer is crafted.
3) An online resource allocation strategy is embedded in NuEMT to minimize the wastage of computational resources on those tasks that are unaligned with the target.
4) A series of experimental studies confirm the effectiveness of NuEMT in reducing agent-environment interaction data requirements in evolutionary RL.

The remainder of this article is organized as follows. Section II contains a brief overview of related work in the literature. Section III contains the preliminaries of ES, EMT, and importance sampling. Section IV describes the generation of auxiliary tasks in RL and presents the NuEMT algorithm. In Section V, we test the algorithm and compare its performance against recent ES baselines. We conclude this article with directions for future research in Section VI.

II. RELATED WORK

There is growing interest in neuroevolutionary algorithms for direct policy search in RL. One of the most notable works in this regard is that by Salimans et al. [8], which established ES as a viable alternative for deep RL. Their experiments revealed the advantages of ES in terms of better exploration capacity (compared to a policy gradient method TRPO [1]) and the ease of scaling to thousands of parallel workers. (This variant of ES has since come to be known as OpenAI-ES in [24] and [25], and will be referred as such hereinafter.) However, even though the runtime of a population-based ES can be greatly reduced by distributing workloads on modern distributed computer systems, the high sample complexity (requiring vast amounts of environment interaction data for policy updates) remains a computational bottleneck.

Earlier works in ES also evinced its potential applicability for direct policy search. In particular, Igel [7] successfully demonstrated the application of the CMA-ES [26] for RL. Two variable metric methods for solving RL tasks, namely, the natural actor critic algorithm [27] and the CMA-ES, were compared and contrasted in [28]. The results showed CMA-ES to be more robust to the selection of hyper-parameters, while being competitive in terms of learning speed.

More recently, significant research effort has been aimed at lowering the sample complexity of ES by introducing better search directions, effective utilization of historical data, and exploration techniques such as novelty search [29]. Choromanski et al. [30] showed that random orthogonal and Quasi Monte Carlo finite difference directions could be more effective for parameter exploration than the random Gaussian directions in [8]. Liu et al. [31] proposed to improve sample efficiency by reducing the variance of the stochastic gradient estimator of the vanilla ES in high-dimensional
optimization. This was done by sampling search directions from a hybrid probabilistic distribution characterized by a gradient subspace—defined by recent historical estimated gradients—and its orthogonal complement. In [10], efficient use of sampled data was sought by an iterative procedure that optimizes a surrogate objective function. A monotonic improvement guarantee for such a procedure was theoretically proven. Further, Conti et al. [32] hybridized novelty search with ES to enhance policy space exploration, encouraging RL agents to exhibit different behaviors that reduce the danger of being stuck indefinitely (hence wastefully) in local optima of deceptive reward functions.

Neuroevolution with genetic algorithms (GAs)—that differ from ES mainly in that they employ crossover operators—has also shown promising results in tuning the parameters of deep neural network policies. Such et al. [33] investigated the use of ES mainly in that they employ crossover operators—has being stuck indefinitely (hence wastefully) in local optima of deceptive reward functions.

III. PRELIMINARIES

In this article, we propose an alternative approach to data-efficient evolutionary RL that reduces expensive agent-environment interaction data requirements. Our method exploits the temporal structure of RL with auxiliary tasks of PELs. Different from [15], our algorithm is the first to augment ES with skills transfer by means of EMT. Preliminaries of these algorithmic components are discussed next.

A. Direct Policy Search

In RL, the goal is to find a policy, i.e., a state-action mapping function, that maximizes cumulative rewards over time of an agent operating in a dynamic and uncertain environment. The policy determines how the agent interacts with the environment. The reward maximization problem can be cast as one of policy parameter optimization as follows:

$$\max_{\theta \in \mathbb{R}^n} F(\pi_\theta)$$  \hspace{1cm} (1)

where the objective (aka fitness) function $F$ is the total returns achieved under policy $\pi_\theta$, parameterized by $\theta$.

There are several methods for solving the maximization problem in (1), such as policy iteration [35], policy gradients [36], or derivative-free optimization [37]. In this work, we focus on derivative-free ES for training neural network policies (where $\theta$ represents the weights of the network), thus allowing $F$ to be treated as a black-box without restriction on the distribution of rewards (sparse or dense), etc.

B. ES for Direct Policy Search

Consider the OpenAI-ES, an ES variant that belongs to the class of natural ESs [8], [38]. The algorithm is based on adding isotropic Gaussian noise of covariance $\sigma^2$ to a mean vector $\hat{\theta}$, transforming (1) to the following maximization of the expected reward (averaged over the induced probability distribution in parameter space)

$$\max_{\hat{\theta} \in \mathbb{R}^n} \mathbb{E}_{\theta \sim \mathcal{N}(\hat{\theta}, \sigma^2)} [F(\pi_\theta)].$$  \hspace{1cm} (2)

This Gaussian-blurred version of the objective function helps to remove nonsmoothness introduced by the environment, hence enabling $\hat{\theta}$ to be effectively updated by the following expected reward gradient (derived by the “log-likelihood trick”)

$$\nabla_{\hat{\theta}} \mathbb{E}_{\theta \sim \mathcal{N}(\hat{\theta}, \sigma^2)} [F(\pi_\theta)] = \frac{1}{\sigma^2} \mathbb{E}_\theta \left[ F(\pi_\theta) (\hat{\theta} - \theta) \right].$$  \hspace{1cm} (3)

In practice, the gradient is approximated via the Monte Carlo method. A fixed number of samples—equivalent to the population size $N$ of the ES algorithm—are drawn from $\mathcal{N}(\theta, \sigma^2 I)$ to compute stochastic gradient estimates of the policy update in every iteration. A pseudocode of the OpenAI-ES is shown in Algorithm 1. The algorithm first perturbs parameter vector $\hat{\theta}$ by sampling $\epsilon_i$’s from a multivariate normal distribution $\mathcal{N}(0, I)$; see steps 3 and 4. The perturbed parameter values $\theta_i = \hat{\theta} + \sigma \epsilon_i$ are then evaluated by running an episode in the environment with the corresponding policy. The results ($F_i$’s) obtained from these episodes approximate the gradient in (3) as $(1/N\sigma^2) \sum_{i=1}^N F_i * (\theta_i - \hat{\theta})$, which is then used to update the mean $\hat{\theta}$. The above repeats until a terminal condition (e.g., function evaluation budget) is met.

A parallel implementation of the OpenAI-ES was discussed in [8] where evaluations were handled independently among distributed workers. The main novelty there was that the algorithm made use of shared random seeds, reducing the bandwidth required for communication between the workers. Such implementations can greatly reduce run time, making ES

Algorithm 1: Pseudocode of the OpenAI-ES

**Input:** $\alpha$: step size, $\sigma$: noise standard deviation, $\hat{\theta}_0$: initial policy parameters, $N$: population size

1: Set $t = 0$
2: repeat
3: Sample $\epsilon_1, \epsilon_2, \ldots, \epsilon_N \sim \mathcal{N}(0, I)$
4: Let $\theta_i = \hat{\theta} + \sigma \epsilon_i$
5: Collect $N$ returns, $F_i = F(\pi_{\theta_i}), \forall i \leq N$
6: Update $\hat{\theta}_{t+1} \leftarrow \hat{\theta}_{t} + \frac{\alpha}{N\sigma^2} \sum_{i=1}^N F_i * (\theta_i - \hat{\theta}_{t})$
7: Set $t = t + 1$
8: until termination condition is met
C. Basics of Evolutionary Multitasking

The motivation behind EMT is to enhance evolutionary search by the exchange and reuse of evolved solutions between jointly optimized tasks. This idea can be applied to boost convergence rates in a difficult target task by solving it in tandem with a group of related auxiliary tasks that are simpler and/or of lower computational cost. Successful applications of this type with EMT have been reported in [20], [39], and [40]. Specifically, the auxiliary tasks serve as informative proxies that quickly guide the target optimization process toward promising regions of the search/parameter space, by the adaptive transfer of discovered solutions.

Let us consider a scenario with $K$ optimization tasks solved simultaneously. Suppose, without loss of generality, each task $T_i \in \{1, 2, \ldots, K\}$, to be a maximization problem instance with search space $\theta_i$ and objective function $F_i : \theta_i \to \mathbb{R}$. Each task may be subject to additional equality and/or inequality constraints. In this setting, the goal of a multitask algorithm is to find in a single run a set of optimal solutions $[\theta_1^*, \theta_2^*, \ldots, \theta_K^*] = \arg \max \{F_1(\theta), F_2(\theta), \ldots, F_K(\theta)\}$, such that $\theta_i^* \in \Theta_i$ and satisfies all constraints of $T_i \forall i$.

In many examples of EMT, the tasks are defined in the same search space [41], i.e., $\Theta_1 = \Theta_2 = \cdots = \Theta_K$, while their objective functions $F_1, F_2, \ldots, F_K$ may differ. (This is also true in the present paper, since all tasks are defined in a common space of neural network policy parameters.) In such cases, we symbolize the single unified space, encompassing all task-specific search spaces, simply as $\Theta$. The unified space provides a shared pathway for distinct but possibly related tasks to exchange mutually beneficial information. For instance, the direct transfer of elite solutions between tasks with correlated objective functions could lead to the rapid discovery of performant solutions. Substantial speedups can thus be achieved in comparison to conventional methods that re-explore search spaces from scratch.

In this article, we consider a special case of EMT where we are primarily interested in solving a target task denoted hereinafter as $T_K$. The remaining $T_1, T_2, \ldots, T_{K-1}$ act as auxiliary tasks catalyzing the evolutionary search.

D. Probabilistic View of Evolutionary Multitasking

Let the target optimization task $T_K$ be

$$\max_{\theta \in \Theta_K} F_K(\theta).$$

Through the lens of probabilistic model-based evolutionary search, we may reformulate (4) as [38]

$$T_K : \max_{p(\theta)} \int_{\Theta_K} F_K(\theta) p_K(\theta) d\theta = \max_{\theta \in \Theta_K} F_K(\theta)$$

where $p_K(\theta)$ represents the probability distribution of an evolving population of candidate solutions. Notice that if $\theta_K^*$ maximizes $F_K(\theta)$, i.e., $F_K(\theta_K^*) = \max_{\theta \in \Theta_K} F_K(\theta)$, then the optimal probability distribution model $p_K^*(\theta)$ that solves (5) is given by $\delta(\theta - \theta_K^*)$, where $\delta(\theta - \theta_K^*)$ is a Dirac delta function centred at $\theta_K^*$. This result follows from the identity:

$$\int_{\Theta_K} F_K(\theta) \delta(\theta - \theta_K^*) d\theta = F_K(\theta_K^*).$$

As such, it is observed that probabilistic reformulation does not change the eventual outcome of the optimization problem.

In the EMT setting, the individual probabilistic models $p_1(\theta), p_2(\theta), \ldots, p_{K-1}(\theta)$ pertaining to $T_1, T_2, \ldots, T_{K-1}$, respectively, are accessible to $T_K$ whilst being jointly optimized in unified space $\Theta$. The individual models encode the skills evolved for the different tasks. Hence, we seek to activate these additional building-blocks of knowledge to accelerate the target search. To that end, we further generalize the probabilistic reformulation of (5) by defining a mixture model (in unified space $\Theta$) as follows:

$$T_K : \max_{w_{K,1},\ldots,w_{K,K},p_K(\theta)} \int_{\Theta} F_K(\theta) \sum_{i=1}^{K} w_{K,i} \cdot p_i(\theta) d\theta$$

where $w_{K,i} \geq 0$, $w_{K,K} > 0$ and $\sum_{i=1}^{K} w_{K,i} = 1$. Equation (6) is the fundamental equation underpinning our proposed NuEMT algorithm (to be fully developed in Section IV). Note that the generalized formulation is optimized if $p_K^*(\theta) = \delta(\theta - \theta_K^*)$ and $w_{K,K} = 1$, implying that the outcome of optimization still remains unchanged under (6). By setting $w_{K,K} = 1$ and $w_{K,i} = 0$, indicating zero intertask transfers, the mixture model simply collapses to $p_K(\theta)$, and we fall back to the well-known form of (5).

However, leveraging the mixture model opens new pathways to actualize skills transfer in EMT, through solution cross-sampling. The mixture coefficient $w_{K,i} \forall i$ essentially reflects the transferability of solutions from a source task $T_i$ to the target $T_K$. Precisely, if candidate solutions evolved for the $i$th task—i.e., drawn from the probabilistic model $p_i(\theta)$—are found to be performant (i.e., return high rewards) for $T_K$ as well, then the value of $w_{K,i}$ can be increased to intensify the cross-sampling of solution prototypes. In contrast, if solutions transferred from a given source do not excel at $T_K$, then the corresponding mixture coefficient can be gradually neutralized. A detailed discussion on this topic can be found in a recent survey [42].

E. Probabilistic Inference by Importance Sampling

Importance sampling is a general statistical technique for inferring properties of a nominal probability distribution $p(\theta)$, given samples drawn from a different distribution $q(\theta)$. The technique is widely used for variance reduction in Monte Carlo methods where $q(\theta)$ takes the form of a biasing distribution from which samples ($\theta$’s) are drawn; an instructive review on the subject can be found in [23], and applications in RL can be referred in [43]. In this article, we shall utilize importance sampling in a unique manner to induce intertask skills transfer in the parameter space of EMT.

Let $E_{p(\theta)}[F_i(\theta)]$ be the expectation of $F_i(\theta)$ under the nominal distribution $p_i(\theta)$ in the parameter space $\Theta_i \subset \mathbb{R}^n$. If the biasing distribution $q(\theta)$ is also defined in $\mathbb{R}^n$ such that $\text{supp}(p_i) \subseteq \text{supp}(q_i)$, where $\text{supp}(p_i) = \{\theta : p_i(\theta) > 0\}$,
then the expectation $E_{p(\theta)}[F_i(\theta)]$ can be reformulated as follows:

$$E_{p(\theta)}[F_i(\theta)] = \int_{\Theta} F_i(\theta)p_i(\theta) \, d\theta = \int_{\Theta} \frac{F_i(\theta)p_i(\theta)}{q_i(\theta)} q_i(\theta) \, d\theta = E_{q_i(\theta)}\left[ \frac{F_i(\theta)p_i(\theta)}{q_i(\theta)} \right]. \quad (7)$$

Here, the multiplicative adjustment to $F_i(\theta)$ given by importance weights $p_i(\theta)/q_i(\theta)$ compensates for sampling from a different distribution $q_i(\theta)$ whilst inferring properties of the nominal distribution $p_i(\theta)$. In the context of EMT (as presented in Section III-D), (7) suggests a technique for updating the probabilistic model $p_k(\theta)$ of target task $T_k$ using solution samples from a different probability distribution [namely, the mixture $\sum_{i=1}^{K} w_{ki} \cdot p_i(\theta)$ in (6)]; hence, leading to the transfer of information through cross-sampling. This new insight lies at the core of the proposed NuEMT algorithm developed in the next section.

IV. NuEMT With Importance Sampling

In this section, we first describe the creation of the auxiliary tasks that catalyze policy search in the target task at hand. Next, we describe the NuEMT algorithm with importance sampling for skills transfer. An adaptive resource allocation strategy is proposed to dynamically adjust computational resources assigned to each constituent task in NuEMT.

A. Construction of Auxiliary Tasks

Skills can be seen as the culmination of a continuous learning process, through accumulating and building on experiences gained from every long or short training session. Bringing this perspective to RL, it is believed that the transfer of skills from simpler tasks (of shorter episode length) could help solve longer and harder tasks more effectively. If tasks with shorter episodes are extracted from the longer and harder task, underlying similarities are expected to exist between them.

In this article, we realize the aforementioned idea for the first time in a multitask setting. To guide the policy search of the target task $T_K$—of maximum episode length “$H$”—we construct a set of auxiliary tasks, $T_1, T_2, \ldots, T_{K-1}$, with shorter episodes of progressive lengths to be solved jointly with $T_K$. The design of the NuEMT algorithm is such that solutions encoding skills evolved in tasks with shorter episodes are progressively transferred to those that are longer and harder. A visualization of the idea is provided in Fig. 1, where there is a total of $K = 3$ tasks, i.e., two auxiliary tasks and one main task. We standardize the episode length of the auxiliary tasks according to their indices, i.e., the $i$th auxiliary task has an episode length of $(i/K) \cdot H$.

Given this setup, the probabilistic formulation for $T_i$ can be written in terms of a mixture model [similarly to (6)] as follows:

$$\max_{p_i(\theta), w_{1i}, \ldots, w_{ki}} E_{q_i(\theta)}[F_i(\pi_\theta)] = \int_{\Theta} F_i(\pi_\theta) q_i(\theta) \, d\theta \quad (8)$$

where $p_i(\theta)$ is the $i$th task-specific probabilistic model, and $q_i(\theta) = \sum_{j=1}^{N_i} w_{ij} \cdot p_j(\theta)$ represents the assimilated mixture model of $T_i$. Note that when $i = 1$, the mixture reduces as $q_1(\theta) = p_1(\theta)$. Hence, no knowledge transfer occurs to the task $T_1$ with the shortest episode length.

B. Naive Stochastic Gradient Estimates

From Section III-B, recall the use of an isotropic multivariate Gaussian, parameterized by its mean $\hat{\theta}$, as the search distribution in the OpenAI-ES. Accordingly, for task $T_i$, we define $p_i(\theta) = N(\hat{\theta}, \sigma^2 I)$. The gradient $\nabla_{\theta_i} \mathbb{E}_{\theta_i}[F_i(\pi_\theta)]$ of the objective function in (8) is

$$\nabla_{\theta_i} \mathbb{E}[F_i(\pi_\theta)] = \nabla_{\theta_i} \int_{\Theta} F_i(\pi_\theta) q_i(\theta) \, d\theta = \sum_{j=1}^{N_i} w_{ij} \int_{\Theta} F_i(\pi_\theta) p_j(\theta) \left[ \nabla_{\theta_i} \log p_j(\theta) \right] \, d\theta \quad (9)$$

where the second statement comes from the log-likelihood trick. Equation (9) yields the Monte Carlo approximation

$$\nabla_{\theta_i} \mathbb{E}[F_i(\pi_\theta)] \approx \frac{1}{N_i} \sum_{i=1}^{N_i} F_i(\pi_{\theta_i})(\theta_k - \hat{\theta}). \quad (10)$$

Here, $\theta_k \sim p_i(\theta)$ and $N_i$ is the total samples (i.e., population size) assigned to task $T_i$. As per (10), it is clear that solutions sampled from probabilistic models of tasks $T_{i \neq i}$ would not directly exert any influence on the stochastic gradient updates of $p_i(\theta)$. In other words, skills transfer between tasks is nonexistent in this naive approach. Therefore, a modification to the gradient estimation is needed for intertask transfers to be established. This can be achieved by means of importance sampling, as disclosed next.

C. Importance Sampling for Skills Transfer

Note that the mixture model $q_i(\theta)$ in (8) includes the task-specific model $p_i(\theta)$ as one of its components. With $w_{ij} > 0$, the condition $\text{supp}(p_i) \subseteq \text{supp}(q_i)$ for importance sampling (see Section III-E) is satisfied. Thus, we can estimate expectations under $p_i(\theta)$—as in (9)—by using samples from the mixture model $q_i(\theta)$ instead. Accordingly, rewriting $p_i(\theta)$ as...
\[
\left[p_i(\theta)/q_i(\theta)\right]q_i(\theta)\right)\] and plugging this into (9), we get
\[
\nabla_{\theta_i} \mathbb{E}[F_i(\pi_{\theta_i})] = w_{i,i} \int_\Theta F_i(\pi_{\theta_i}) p_i(\theta) \left[ \nabla_{\theta_i} \log p_i(\theta) \right] q_i(\theta) \, d\theta
\]
\[
\approx \frac{w_{i,i}}{\sigma^2} \sum_{k=1}^{N_i} F_i(\pi_{\theta_k}) \left[ \sum_{l=1}^{i-1} w_{l,i} \cdot p_i(\theta_l) \right] (\theta_k - \tilde{\theta}_i)
\]
(11)
where \(\theta_k \sim q_i(\theta)\). As a result of the reformulation, solutions sampled from all components of the mixture model \(q_i(\theta)\) shall directly influence stochastic gradient updates of the search distribution parameter \(\tilde{\theta}_i\) of \(T_i\). Hence, through importance sampling, the transfer of solution prototypes encoding evolved skills is activated from tasks of shorter episode lengths to those that are longer and harder.

In addition to \(\tilde{\theta}_i\), the mixture coefficients of \(q_i(\theta)\) are also updated during the search process. Applying importance sampling again, the gradient estimate with respect to \(w_{i,j} \ (j \leq i)\) can be obtained as follows:
\[
\frac{\partial \mathbb{E}[F_i(\pi_{\theta_i})]}{\partial w_{i,j}} = \int_\Theta F_i(\pi_{\theta_i}) p_i(\theta) \, d\theta
\]
\[
= \int_\Theta F_i(\pi_{\theta_i}) p_i(\theta) q_i(\theta) \, d\theta
\]
\[
\approx \frac{1}{N_i} \sum_{k=1}^{N_i} F_i(\pi_{\theta_k}) \sum_{l=1}^{i} w_{l,i} \cdot p_i(\theta_l).
\]
(12)

D. Derived NuEMT Update Equations

In order to make the methodology robust and invariant to outliers as well as arbitrary yet order-preserving fitness transformations, a standard rank-based fitness shaping function defined by utility values \(u_i\) as substitute to the actual returns \(F_i\) is considered [38]
\[
u_{i,k} = \max\left(0, \log\left(\frac{N_i}{2} + 1\right) - \log k\right)
\]
where \(u_{i,k}\) is the utility of the \(k\)th sample in a sorted population list, i.e., \(F_i(\pi_{\theta_1}) \geq F_i(\pi_{\theta_2}) \geq \cdots \geq F_i(\pi_{\theta_{N_i}}) \geq \cdots \geq F_i(\pi_{\theta_N})\), in the population of \(T_i\).

In practice, if the gap between \(\tilde{\theta}_j\) and \(\tilde{\theta}_i\) becomes large, the importance weight \([p_i(\theta_k)/q_i(\theta_k)]\) in (11) rapidly approaches zero for \(\theta_k \sim p_i(\theta)\) due to distribution sparsity in even moderately high-dimensional parameter spaces. The importance weights thus suppress the influence of cross-sampled solutions on the update of \(\tilde{\theta}_i\). To resolve this issue, we adopt a projection technique introduced in [44]; accordingly, solutions \(\theta_k \sim p_i(\theta)\) that lie outside Mahalanobis distance \(r\) from \(p_i(\theta)\) are mapped back to a distance \(r\) (set to 1 in all experiments) while maintaining the same directional bias. A mapped solution, \(\theta_k'\), is defined as follows:
\[
\theta_k' = \begin{cases} 
\tilde{\theta}_i + (\theta_k - \tilde{\theta}_i) \cdot \min\left(1, \frac{r}{\sqrt{\frac{2\pi}{\sigma^2}}} \right), & \text{if } \theta_k \neq p_i(\theta) \\
\theta_k, & \text{otherwise}
\end{cases}
\]
(14)
with its utility value simply retained as \(u_{i,k}(\pi_{\theta_k'}) = u_{i,k}(\pi_{\theta_k})\).

Based on the above, the final update equation for parameter \(\tilde{\theta}_i\) is given by
\[
\tilde{\theta}_i \leftarrow \tilde{\theta}_i + \frac{\alpha}{N_i} \sum_{k=1}^{N_i} u_{i,k} \cdot \frac{p_i(\theta_k')}{q_i(\theta_k')} (\theta_k' - \tilde{\theta}_i)
\]
(15)
where \(\alpha\) is the learning stepsize for \(\tilde{\theta}_i\).

For the update of the mixture coefficients, the constraints \(\sum_{j=1}^{N_j} w_{i,j} = 1 \text{ and } w_{i,j} \geq 0\) must be satisfied. Thus, an additional step projecting the gradient approximation of (12) to the constraint plane \(C\) is needed. Denoting the normal vector to the plane as \(\vec{c} = [1, \ldots, 1]\), and the gradient estimate as \(\vec{b} = \left[(\partial \mathbb{E}[F_i(\pi_{\theta_i})])/(\partial w_{i,1}), \left(\partial \mathbb{E}[F_i(\pi_{\theta_i})])/(\partial w_{i,2}), \ldots, \left(\partial \mathbb{E}[F_i(\pi_{\theta_i})])/(\partial w_{i,i})\right]\), the scaled projection of the gradient on the plane takes the form
\[
\text{proj}_C(\vec{b}) = \beta \cdot \left(\vec{c} - \frac{\vec{b} \cdot \vec{c}}{\sum_{j=1}^{i} \frac{\partial \mathbb{E}[F_i(\pi_{\theta_i})]}{\partial w_{i,j}}} \right)
\]
(16)
where \(\beta\) is the learning stepsize for the mixture coefficients. Then, the final update equation for \(w_i = [w_{i,1}, w_{i,2}, \ldots, w_{i,i}]\) takes the following form:
\[
w_i \leftarrow w_i + \lambda \cdot \text{proj}_C(\vec{b})
\]
(17)
where \(\lambda > 0\) is a scaling factor to ensure nonnegativity. It can be shown that the coefficients converge to steady values under (17). Assume (for simplicity) the individual probabilistic models \(p_1(\theta), p_2(\theta), \ldots, p_l(\theta)\) pertaining to \(T_1, T_2, \ldots, T_l\) have converged to the optimal Dirac delta functions \(\delta(\theta - \theta^*_1), \delta(\theta - \theta^*_2), \ldots, \delta(\theta - \theta^*_l)\), respectively, with \(\theta^*_i \neq \theta^*_j \forall i \neq j\). From the first line of (12), it follows that \((\partial \mathbb{E}[F_i(\pi_{\theta_i})])/(\partial w_{i,i}) \gg (\partial \mathbb{E}[F_i(\pi_{\theta_i})])/(\partial w_{i,j})\), since \(\int_\Theta F_i(\theta) \delta(\theta - \theta^*_i) \, d\theta > \int_\Theta F_i(\theta) \delta(\theta - \theta^*_i) \, d\theta\). Plugging the inequality into (16), the \(i\)th component of \(\text{proj}_C(\vec{b})\) is
\[
\beta \left(\frac{\partial \mathbb{E}[F_i(\pi_{\theta_i})]}{\partial w_{i,i}} - \frac{1}{i} \sum_{j=1}^{i} \frac{\partial \mathbb{E}[F_i(\pi_{\theta_i})]}{\partial w_{i,j}}\right) > 0
\]
(18)
This shows that the mixture coefficient \(w_{ij}\) tends to increase under the update. What’s more, the scaling factor \(\lambda\) in (17) serves to ensure \(w_{ij} \leq 1\). Hence, \(w_{ij} \forall i\), gradually converges to a steady value of 1, while all other \(w_{ij}\)’s go to 0. We demonstrate this behavior through an experimental result in Section V-D of this article.

E. Adaptive Resource Allocation Strategy

Learning a high mixture coefficient \(w_{K,j}\) suggests strong transferability between \(T_K\) and \(T_j\). The auxiliary task \(T_j\) could then serve as a cheap proxy for the main task \(T_K\) for quickly progressing the target search. Greater computational resources can be allocated to \(T_j\), while facilitating the transfer of evolved skills to \(T_K\) at a fraction of the cost. Such a resource allocation strategy has already been demonstrated to work well in the context of EMT for evolutionary machine learning [20]. With that in mind, we incorporate an adaptive resource allocation strategy into NuEMT, assigning population sizes for the evolution of the auxiliary tasks as follows:
\[
N_j = N_{\text{total}} \cdot w_{K,j}
\]
(19)
where \( N_{\text{total}} \) is the total population size of the NuEMT and \( N_j \) is the size allocated to \( T_j \). Equation (19) reduces resource wastage, since an auxiliary task uncorrelated with the target will get a small or zero population size.

F. Summarizing NuEMT for RL

Tying together the derived update equations, we herein summarize the NuEMT algorithm for RL. Our methodology incorporates mixture modeling as a means of intertask relationship capture, to control the extent of skills transfer between tasks. Auxiliary task \( T_i \), for \( i = 1, 2, \ldots, K - 1 \), is extracted from the same environment as the main task \( T_K \), but has a shorter episode length, i.e., \( (i/K) \cdot H \). A pseudocode of the overall procedure is given in Algorithm 2.

At initialization, all components of the mixture models are uniformly weighted by setting \( w_{i,j} = 1/i \) for \( j = 1, 2, \ldots, i \). An equal population size of \( N_i = (N_{\text{total}}/K) \) is allocated to each task. In each iteration, every task samples solutions, \( \theta_1, \theta_2, \ldots, \theta_{N_i} \) from its mixture model \( q_i(\theta) \). Each \( \theta_k \) represents a policy \( \pi_{\theta_k} \), and \( F_i(\pi_{\theta_k}) \) represents the total reward received from an episode run with length \( H_i = (i/K) \cdot H \). The \( N_i \) reward values received are then used for parameter updates as formulated in (15) and (17). Subsequently, the coefficients \( \omega_k \), from the main task’s mixture model \( q_K(\theta) \) are used to determine the population size to be allocated to each of the auxiliary tasks in the next iteration. This process continues until a terminal condition is met.

In our implementation, we find it useful to perform state normalization [37] as it enables different state components to have a fair share of influence during training. A similar normalization approach known as virtual batch normalization is also used by OpenAI-ES [8]. In addition, weight decay is added as a form of regularization to prevent parameters of the policy network from exploding. Finally, we adopt mirror sampling [45] as a variance reduction technique.

It is worth noting that when we compare NuEMT with a conventional neuroevolutionary algorithm, the computational cost of NuEMT (per iteration) will be lower given the same \( N_{\text{total}} \). This is because each of the \( K \) tasks is assigned \( (N_{\text{total}}/K) \) solutions to start with, and, assuming the computational cost of evaluating a solution for \( T_i \) to be \( C_i \), the total solution will be \( \sum_i C_i \cdot (N_{\text{total}}/K) < N_{\text{total}} \cdot C_K \) since \( C_1 < C_2 < \cdots < C_K \). This is especially crucial for RL problems that may be dealing with extremely long episodes in large-scale simulations.

V. EXPERIMENTAL STUDIES

In this section, we present a set of experiments on continuous control tasks from the OpenAI Gym [46] (see Figs. 2 and 3) to showcase the efficiency of the NuEMT algorithm.

A. Experimental Configuration

In our experiments, we compare against the OpenAI-ES [8] and the recently proposed PEL framework [15]. Our implementation of the latter uses the OpenAI-ES as the base optimizer and is referred to as PEL for the rest of the section.

Algorithm 2 Pseudocode of the NuEMT Algorithm

**Input:** \( N_{\text{total}}, K \): population size, \( K \): number of tasks, \( \sigma \): standard deviation, \( \alpha \): stepsize, \( \beta \): mixture stepsize, \( H \): full time horizon, \( T = \{T_1, T_2, \ldots, T_K\} \): set of tasks

1: Set mixture coefficient \( w_{i,j} = 1/i \), \( \forall j \leq i \) for \( T_i \), \( \forall i \)
2: Set population size \( N_i = w_{K,i} \cdot N_{\text{total}} \) for \( T_i \), \( \forall i \)
3: Set \( H_i = \frac{H}{K} \cdot H, \forall i \leq K \)
4: Set \( \theta_0 = 0 \in \mathbb{R}^n \) for \( T_i \), \( \forall i \)
5: repeat
6: for each \( T_i \) in \( \mathcal{D} \) do
7: Set \( q_i(\theta) = \sum_{j \leq i} w_{i,j} \cdot p_j(\theta) \), where \( p_j(\theta) = N(\bar{\theta}_j, \sigma^2 I) \)
8: Sample \( \theta_1, \theta_2, \ldots, \theta_{N_i} \sim q_i(\theta) \)
9: Collect \( N_i \) returns based on episode length \( H_i \):
10: \[ F_{i,k} = F_i(\pi_{\theta_k}) \quad \forall k \leq N_i \]
11: Sort \( F_{i,k} \) in descending order of fitness, and apply rank-based fitness shaping:
12: \[ u_{i,1} \geq \cdots \geq u_{i,N_i} \leftarrow F_{i,1} \geq \cdots \geq F_{i,N_i} \]
13: Project solutions not sampled from \( p_i(\theta) \):
14: \[ \hat{\theta}_k = \begin{cases} \bar{\theta}_k + (\bar{\theta}_k - \hat{\theta}_k) & \text{if } \theta_k \neq p_i(\theta) \\ \hat{\theta}_k & \text{otherwise} \end{cases} \]
15: Calculate gradient estimates for \( \hat{\theta}_i \) and \( w_{i,j} \):
16: \[ \nabla \hat{\theta}_i = \frac{1}{N_i} \sum_{k=1}^{N_i} u_{i,k} \cdot \frac{p_j(\theta_k)}{q_i(\theta_k)} (\hat{\theta}_k - \bar{\theta}_i) \]
17: Perform update step:
18: \[ \hat{\theta}_i \leftarrow \hat{\theta}_i + \alpha \cdot \nabla \hat{\theta}_i \]
19: \[ w_i \leftarrow w_i + \lambda \cdot \mathsf{proj}_{C}(\{\nabla w_{i,1}, \nabla w_{i,2}, \ldots, \nabla w_{i,j}\}) \]
20: Update \( N_i \leftarrow w_{K,i} \cdot N_{\text{total}}, \forall i \leq K \)
21: until termination condition is met

The comparison with the OpenAI-ES allows us to investigate the sample efficiency of our multitask algorithm alongside its single-task counterpart. Similarly, the comparison between sequential transfer (PEL) and multitask transfer (NuEMT) helps us to understand the differences in performance between the two approaches across a variety of environments. This is especially important since we are interested in observing how the limitations of sequential transfers can be averted by multitasking. Recall, the notion of sequential transfer may degrade or stagnate performance if poor or overspecialized solutions are propagated from the simpler to the harder tasks.
The experimental setups are configured as follows. The episode length of the main target task in NuEMT is equal to that of the final task in the PEL baseline; this is equal to the episode length of the single-task in the OpenAI-ES. The number of tasks in NuEMT and PEL are kept the same, for fairness of comparison. For instance, let us assume that there are three tasks in NuEMT and PEL, and the full episode length is 1200 timesteps. In the case of NuEMT, the first and second auxiliary tasks will have episode lengths of 400 and 800 timesteps, respectively, while the target task will have a maximal episode length of 1200 timesteps. Similarly, for PEL, the episode scheduler is configured as follows. The first task will have an episode length of 400 timesteps, followed by 800 timesteps for the second task, and 1200 timesteps for the last task. In contrast, the single-task OpenAI-ES will have a constant episode length of 1200 timesteps for evaluating all policy parameters generated during its evolutionary run.

A single run of an algorithm ends when the total number of agent-environment interaction timesteps performed (summed across all tasks in PEL and NuEMT) exceeds a predefined termination condition. This termination condition also determines manual settings of the time scheduler of PEL, i.e., the number of agent-environment interactions performed in each task before moving on to the next. In our implementation, we take the total timesteps for each task in PEL to be uniform (obtained by dividing the termination condition by \( K \)). Note, we use total timesteps instead of the actual wall-clock time used in [15].

The total population size \( N_{\text{total}} \) for all algorithms is the same. In NuEMT, a minimum population size of \( N_{\text{total}} / K \) is imposed for the main target task along with the adaptive resource allocation strategy, to prevent its population from collapsing—as an exceedingly small population may lead to brittle performance with high variance. For PEL, the population size remains the same for all tasks. The compared algorithms train policies with identical architectures, namely, multilayer perceptrons with two hidden layers of 64 nodes and tanh activation functions.

For our implementation, we apply the same parallelization approach in [37] using the Python library Ray [47]. All experiments are performed on a single machine with 12-core/24-thread CPU. Each worker holds unique random seeds for sampling noise in the shared noise table as well as initializing the OpenAI Gym environment.

### B. Results on Advanced Physics Simulation MuJoCo Tasks

Here, we compare NuEMT against the baseline algorithms on a variety of MuJoCo tasks. We selected 6 of the popular simulations commonly used in the RL literature, as depicted in Fig. 2. For the Humanoid-v2 tasks, we find that the survival bonus from the reward function encourages policies that make the MuJoCo models stand at the same spot until maximum episode length is reached [37], resulting in getting stuck in a local optima of the policy space. To resolve this issue, we minus off the survival bonus (score of 5) from the reward function at each timestep during training. Table I presents the details of our experiments for each MuJoCo tasks. Our experiment also conducted each simulation for a total of 20 independent trials. In every trial, different random seeds are assigned to each worker and the Gym environment.

The results shown in Table II are the mean and standard deviation of the total rewards achieved by all the algorithms at different timesteps. The convergence trends of each algorithm are also shown in Fig. 4. We see from Table II that NuEMT outperforms the comparative algorithms at different timesteps for most of the control tasks. Comparing NuEMT with OpenAI-ES, the former is found to offer significant speedup. Note, the main difference between the two algorithms is the transfer of evolved skills in NuEMT. Moreover, the multitask strategy of NuEMT enables accelerated convergence on 5 out of the 6 Mujoco tasks compared to the sequential PEL. The convergence plots in Fig. 4 reveal a similar story. This is especially clear when we observe convergence behaviors at the initial stages of evolution, where the proposed algorithm rapidly attains higher rewards (on the main task) than...
the baseline algorithms. Note, on Swimmer-v2, PEL fails to outperform the OpenAI-ES. In contrast, the convergence trends of NuEMT consistently provide strong evidence of its ability to achieve lean evolutionary RL (measured in terms of the amount of agent-environment interaction data needed).

C. Results on Box2d Simulators

Here, we selected BipedalWalker as a representative in the Box2D simulator from OpenAI Gym. In the BipedalWalker experiment, the robot gets up to 300 plus points when it reaches the far end and 100 points are deducted if the robot falls. We have adjusted the length of the legs (depicted in Fig. 3) such that each continuous control task will have different stability issues to overcome while traversing the terrain. Table III shows the number of tasks, episode length, number of timesteps (termination condition), and other implementation details for our experiments. For each simulation, a total of 20 independent trials are conducted. Similar to the MuJoCo experiments, unique random seeds are assigned to each worker and the Gym environment in every trial.

Table IV shows the mean and standard deviations of the total rewards at different timesteps. As seen from the results, NuEMT outperforms the comparative algorithms at most

| Simulation       | No. of Timesteps | NuEMT Mean ± Std | OpenAI-ES Mean ± Std | PEL Mean ± Std |
|------------------|------------------|------------------|---------------------|---------------|
| HalfCheetah-v2   | 2,500,000        | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
|                  | 5,000,000        | 2200.48 ± 182.75 | 2243.82 ± 149.75    | 2323.64 ± 164.25 |
| Swimmer-v2       | 2,500,000        | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
|                  | 5,000,000        | 2200.48 ± 182.75 | 2243.82 ± 149.75    | 2323.64 ± 164.25 |
| Hopper-v2        | 8,000,000        | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
|                  | 16,000,000       | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
| Ant-v2           | 4,500,000        | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
|                  | 9,000,000        | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
| Walker2d-v2      | 25,000,000       | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
|                  | 50,000,000       | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
| Humanoid-v2*     | 10,000,000       | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
|                  | 20,000,000       | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
|                  | 30,000,000       | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
|                  | 40,000,000       | 2176.50 ± 182.75 | 2217.82 ± 149.75    | 2303.64 ± 164.25 |
Fig. 5. Historical update trends of the mixture coefficients for the main and auxiliary tasks in the Hopper-v2 environment. The shaded area represents one standard deviation on either side of the mean. The convergence trends align well with the conclusion drawn at the end of Section IV-D.

timesteps in all three simulations. Comparing NuEMT with the base optimizer, OpenAI-ES, the former is found to be significantly more data efficient. This improvement is a consequence of intertask skills transfer, which is the fundamental algorithmic distinction between NuEMT and OpenAI-ES. The superior sample efficiency is achieved by tapping on useful information from simpler tasks (of shorter agent-environment interaction episodes) to quickly achieve better performance on the longer and harder task at hand.

While the PEL baseline performs slightly better than NuEMT on the $0.5 \times$ leg length BipedalWalker-v3 simulation, it struggles to perform consistently in the other two simulations that pose increasing difficulty in maintaining balance due to longer leg lengths. In contrast, the NuEMT methodology maintains sample efficient performance in all three experiments, vastly outperforming competitors on $1.0 \times$ and $1.5 \times$ leg length.

D. Analysing the Behavior of NuEMT

1) Update History of the Mixture Coefficients: During the initial stages of evolution, if task $T_i$ transfers useful skills to the main task $T_K$, we expect the mixture coefficient $w_{K,i}$ to increase, allocating greater computational resources to $T_i$ to quickly discover better solutions at lower cost. However, at later stages, after the target probabilistic model $p_K(\theta)$ has arrived at a promising region of the search space, the mixture coefficient value $w_{K,i}$ is expected to increase (whereas $w_{K,i}$ is expected to drop), thus increasing computational effort on refinement of the target search. To verify this intuition, we perform 20 independent trials in the Hopper-v2 environment (for an extended period of 40 million timesteps) to investigate the temporal latent relationship between the main task and the auxiliary task by observing the historical updates of the learnt mixture coefficients. Averaged convergence plots are shown in Fig. 5, and are found to not only align qualitatively with our intuition, but also with the theoretical conclusion drawn at the end of Section IV-D.

Fig. 6. Performance of NuEMT with learnt mixture coefficients versus NuEMT with fixed mixture coefficients. Shaded area represents one standard deviation on either side of the mean. (a) HalfCheetah-v2. (b) Hopper-v2.

2) Effectiveness of Mixture Coefficient Learning: It is worth understanding how our algorithm would perform with and without mixture coefficient learning. To this end, we perform two sets of experiments on HalfCheetah-v2 and Hopper-v2, two of the MuJoCo simulations. One experiment is performed with our mixture coefficient learning and subsequent mixture coefficient update, while the other experiment is performed with a fixed set of mixture coefficients that does not change over iterations. The number of auxiliary tasks is set to 1, as per the configuration in earlier results. The mean performance between the two sets of experiment is shown in Fig. 6. We see that mixture coefficient learning in (17) is able to uplift the performance of the fixed weight variant by up to 23% at the end of the training curve. This provides an interesting outlook of how mixture coefficients can directly affect the gradient estimate for $\tilde{\theta}$ [see (15)], driving search toward good solutions by self-adapting the sampling between different search distribution models.

3) Analyzing Effects of the Number of Auxiliary Tasks: Here, we investigate how the number of auxiliary tasks in
TABLE III
DETAILS OF THE BOX2D SIMULATORS

| Simulation          | Algorithm | No. of Tasks | Episode Length | Termination Condition (No. of Timesteps) | Population Size | Learning Parameters |
|---------------------|-----------|--------------|----------------|------------------------------------------|----------------|---------------------|
| BipedalWalker-v3    | NuEMT     | 4            | $T_1: 400$, $T_2: 800$, $T_3: 1200$, $T_4: 1600$ | 70,000,000              | 128            | $\alpha = 0.05$, $\beta = 0.05$ |
|                     | PEL       |              |                |                                          |                | $\alpha = 0.05$     |
|                     | OpenAI-ES | 1            | 1600           |                                          |                | $\alpha = 0.05$, $\beta = 0.05$ |
| BipedalWalker-v3    | NuEMT     | 4            | $T_1: 400$, $T_2: 800$, $T_3: 1200$, $T_4: 1600$ | 80,000,000              | 128            | $\alpha = 0.05$, $\beta = 0.05$ |
|                     | PEL       |              |                |                                          |                | $\alpha = 0.05$     |
|                     | OpenAI-ES | 1            | 1600           |                                          |                | $\alpha = 0.05$, $\beta = 0.05$ |
| BipedalWalker-v3    | NuEMT     | 4            | $T_1: 400$, $T_2: 800$, $T_3: 1200$, $T_4: 1600$ | 90,000,000              | 128            | $\alpha = 0.05$, $\beta = 0.05$ |
|                     | PEL       |              |                |                                          |                | $\alpha = 0.05$     |
|                     | OpenAI-ES | 1            | 1600           |                                          |                | $\alpha = 0.05$     |

**Bipedal walkers with varying leg lengths are shown in Figure 3.**

TABLE IV
COMPARISON OF THE MEAN PERFORMANCE AND STANDARD DEVIATION ACROSS BOX2D TASKS AND COMPARATIVE ALGORITHMS. RESULTS IN BOLD INDICATE BEST MEAN PERFORMANCE

| Simulation          | No. of Timesteps | NuEMT Mean ± Std | OpenAI-ES Mean ± Std | PEL Mean ± Std |
|---------------------|------------------|-------------------|----------------------|----------------|
| BipedalWalker-v3    | 17,500,000       | 282.92 ± 38.13    | 58.81 ± 89.33        | 67.27 ± 23.20  |
| (0.5x Leg Length)   | 35,000,000       | 295.55 ± 19.09    | 100.70 ± 120.17      | 206.39 ± 51.93 |
|                     | 52,500,000       | 296.13 ± 14.34    | 125.03 ± 126.98      | 302.68 ± 40.02 |
|                     | 70,000,000       | 295.11 ± 12.91    | 181.70 ± 118.19      | 331.06 ± 13.60 |
| BipedalWalker-v3    | 20,000,000       | 233.17 ± 74.27    | 5.18 ± 3.37          | 8.61 ± 0.66    |
| (1.0x Leg Length)   | 40,000,000       | 262.94 ± 73.06    | 6.47 ± 2.45          | 11.31 ± 8.91   |
|                     | 60,000,000       | 290.61 ± 19.22    | 7.85 ± 3.24          | 13.92 ± 12.29  |
|                     | 80,000,000       | 297.38 ± 15.47    | 8.15 ± 3.64          | 44.64 ± 80.10  |
| BipedalWalker-v3    | 22,500,000       | 27.72 ± 66.12     | 10.09 ± 4.67         | 13.34 ± 0.17   |
| (1.5x Leg Length)   | 45,000,000       | 50.43 ± 115.70    | 11.35 ± 4.53         | 13.03 ± 0.36   |
|                     | 67,500,000       | 133.33 ± 128.44   | 12.46 ± 0.415        | 12.83 ± 0.53   |
|                     | 90,000,000       | 152.28 ± 127.35   | 12.79 ± 0.52         | 12.68 ± 0.57   |

NuEMT affects its performance. We ran several experiments with different numbers of auxiliary tasks in the BipedalWalker-v3 simulation. The BipedalWalker-v3 simulation used has the default leg length. For the experimental settings, each experiment consists of 20 independent trials with 20 different seeds. The population size is set to be 128 candidate solutions. The learning rates $\alpha, \beta$ for parameter $\theta$ and mixture coefficient $w$ are fixed as 0.05 and 0.05, respectively. Our goal is to analyze the effect of increasing the number of tasks given a fixed population size. Averaged results are shown in Fig. 7.

From Fig. 7, we see that NuEMT with three auxiliary tasks performs the best among the five line plots, followed by NuEMT with four auxiliary tasks, NuEMT with two auxiliary tasks, NuEMT with one auxiliary task, and finally NuEMT with no auxiliary task (which reduces to the OpenAI-ES). While having more auxiliary tasks could provide more information to the target, the total population size must also grow to allow the search on shorter episode lengths to generate useful transferrable skills. Hence, with a fixed population size of 128 solutions, NuEMT with three auxiliary tasks managed to outperform NuEMT with four auxiliary tasks in the same environment.

VI. CONCLUSION
In this article, we explored the application of EMT as a novel means to achieve data-efficient evolutionary RL. Our proposed NuEMT algorithm is based on the idea of harnessing...
useful information (transferrable skills) from auxiliary tasks with shorter episode lengths, to quickly optimize a neural network policy for the target task at hand. The uniqueness of NuEMT lies in utilizing the statistical importance sampling technique as the information transfer mechanism within the base optimizer, OpenAI-ES, without having to modify its other search operators. The multitasking trick is shown to provide enhanced sample efficiency, attaining higher cumulative rewards with lesser agent-environment interactions.

In our experiments, a variety of continuous control environments from the OpenAI Gym were considered. The results unveiled significant advantages of multitasking over the single-task OpenAI-ES as well as a sequential transfer-based ES (which made use of the same auxiliary tasks). Multitasking overcomes the threat faced by sequential transfer in those cases where solutions evolved for shorter episodes do not propagate well to longer and harder tasks in the future. Our results thus mark a major step forward in confirming the viability of evolutionary algorithms as simple, scalable, and sample-efficient alternatives for deep RL.

For the next step in this line of research, we plan to extend the general idea of leveraging simpler tasks to improve learning on complex problems beyond the realms of RL. Other machine learning subfields, such as neural architecture search, may also benefit greatly from the potential to jointly evolve multiple tasks, producing diverse models specialized to different datasets and/or different hardware constraints in a single evolutionary run.

REFERENCES

[1] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, “Trust region policy optimization,” in Proc. 32nd Int. Conf. Mach. Learn., vol. 37, 2015, pp. 1889–1897.

[2] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” in Proc. 4th Int. Conf. Learn. Represent., 2016, pp. 1–14.

[3] G. Barth-Maron et al., “Distributed distributional deterministic policy gradients,” in Proc. 6th Int. Conf. Learn. Represent., 2018, pp. 1–16.

[4] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.

[6] D. Silver et al., “Mastering the game of go with deep neural networks and tree search,” Nature, vol. 529, pp. 484–503, Jan. 2016.

[7] C. Igel, “Neuroevolution for reinforcement learning using evolution strategies,” in Proc. Conge. Evol. Comput., vol. 4, 2003, pp. 2588–2595.

[8] T. Salimans, J. Ho, X. Chen, and I. Sutskever, “Evolution strategies as a scalable alternative to reinforcement learning,” 2017, arXiv:1703.03864.

[9] S. Risi and J. Kogelius, “Neuroevolution in games: State of the art and open challenges,” IEEE Trans. Comput. Intell. Al Games, vol. 9, no. 1, pp. 25–41, Mar. 2017.

[10] G. Liu et al., “Trust region evolution strategies,” in Proc. AAMAI Conf. Artif. Intell., vol. 33, 2019, pp. 4352–4359.

[11] T. Gangwani and J. Peng, “Policy optimization by genetic distillation,” in Proc. 6th Int. Conf. Learn. Represent., 2018, pp. 1–16.

[12] Y. Yu, “Towards sample efficient reinforcement learning,” in Proc. 27th Int. Joint Conf. Artif. Intell., 2018, pp. 5739–5743.

[13] F. Zhuang et al., “A comprehensive survey on transfer learning,” Proc. IEEE, vol. 109, no. 1, pp. 43–76, Jan. 2021.

[14] A. Gupta, Y. Ong, and L. Feng, “Insights on transfer optimization: Because experience is the best teacher,” IEEE Trans. Emerg. Topics Comput. Intell., vol. 2, no. 1, pp. 51–64, Feb. 2018.

[15] L. Fuks, N. H. Awad, F. Hutter, and M. Lindauer, “An evolution strategy with progressive episode lengths for playing games,” in Proc. 28th Int. Joint Conf. Artif. Intell., 2019, pp. 1234–1240.

[16] P. Chrabaszcz, I. Loschilov, and F. Hutter, “Back to basics: Benchmarking canonical evolution strategies for playing Atari,” in Proc. 27th Int. Joint Conf. Artif. Intell., 2018, pp. 1419–1426.

[17] A. Gupta, Y. Ong, and L. Feng, “Multifactorial evolution: Toward evolution multi-tasking,” IEEE Trans. Evol. Comput., vol. 20, no. 3, pp. 343–357, Jun. 2016.

[18] L. Zhang, Y. Xie, I. Chu, L. Feng, C. Chen, and K. Liu, “A study on multi-form multi-objective evolutionary optimization,” Memet. Comput., vol. 13, no. 3, pp. 307–318, 2021.

[19] K. K. Bai, Y. 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, Feb. 2020.

[20] K. K. Bai, A. Gupta, Y.-S. Ong, and P. S. Tan, “Cognizant multitasking in multiobjective multifactorial evolution: MO-MFEA-II,” IEEE Trans. Cybern., vol. 51, no. 4, pp. 1784–1796, Apr. 2021.

[21] Q. Wang, Y. Huang, Y. Wang, M. Li, and L. Feng, “Solving vehicle routing problem by memetic search with evolutionary multitasking,” Memet. Comput., vol. 14, no. 1, pp. 31–44, 2022.

[22] S. T. Tokdar and R. E. Kass, “Importance sampling: A review,” Wiley Interdiscip. Rev. Stat. Comput., vol. 2, no. 1, pp. 54–60, 2010.

[23] Q. Shang, Y. Huang, Y. Wang, M. Li, and L. Feng, “Solving vehicle routing problem by memetic search with evolutionary multitasking,” Memet. Comput., vol. 14, no. 1, pp. 31–44, 2022.

[24] S. Risi and J. Togelius, “Neuroevolution in games: State of the art and open challenges,” IEEE Trans. Comput. Intell. AI Games, vol. 9, no. 1, pp. 43–76, Jan. 2021.

[25] J. Schmidhuber, “Natural evolution strategies,” J. Mach. Learn. Res., vol. 10, no. 1, pp. 1179–1211, Aug. 2022.

[26] D. Silver et al., “Asynchronous methods for deep reinforcement learning,” in Proc. 33rd Int. Conf. Mach. Learn., vol. 48, 2016, pp. 1928–1937.

[27] D. P. Bertsekas, Dynamic Programming and Optimal Control, 3rd ed. Belmont, MA, USA: Athena Sci., 2005.

[28] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in Proc. Int. Conf. Mach. Learn., 2014, pp. 387–395.

[29] H. Mania, A. Guy, and B. Recht, “Simple random search of static linear policies is competitive for reinforcement learning,” in Advances in Neural Information Processing Systems, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2018.

[30] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, “Natural evolution strategies,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 949–980, 2014.

[31] J. Ding, C. Yang, Y. Jin, and T. Chai, “Generalized multitasking for evolutionary optimization of expensive problems,” IEEE Trans. Evol. Comput., vol. 23, no. 1, pp. 44–58, Feb. 2019.
[41] Y. Ong and A. Gupta, “Evolutionary multitasking: A computer science view of cognitive multitasking,” Cogn. Comput., vol. 8, no. 2, pp. 125–142, 2016.

[42] A. Gupta, L. Zhou, Y.-S. Ong, Z. Chen, and Y. Hou, “Half a dozen real-world applications of evolutionary multitasking, and more,” IEEE Comput. Intell. Mag., vol. 17, no. 2, pp. 49–66, May 2022.

[43] J. P. Hanna, S. Niekum, and P. Stone, “Importance sampling in reinforcement learning with an estimated behavior policy,” Mach. Learn., vol. 110, pp. 1–51, May 2021.

[44] J. C. Wong, A. Gupta, and Y.-S. Ong, “Can transfer neuroevolution tractably solve your differential equations?” IEEE Comput. Intell. Mag., vol. 16, no. 2, pp. 14–30, May 2021.

[45] D. Brockhoff, A. Auger, N. Hansen, D. V. Arnold, and T. Hohm, “Mirrored sampling and sequential selection for evolution strategies,” in Parallel Problem Solving from Nature (Lecture Notes in Computer Science 6238). Berlin, Germany: Springer, 2010, pp. 11–21. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-15844-5_2#citeas

[46] G. Brockman et al., “OpenAI gym,” 2016, arXiv:1606.01540.

[47] P. Moritz et al., “Ray: A distributed framework for emerging AI applications,” in Proc. 13th USENIX Symp. Oper. Syst. Design Implement., 2018, pp. 561–577.

Nick Zhang received the B.Sc. degree in mathematical sciences from Nanyang Technological University, Singapore, in 2017, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. His current research interests include evolutionary machine learning, multitask optimization, and reinforcement learning.

Abhishek Gupta (Senior Member, IEEE) received the Ph.D. degree in engineering science from the University of Auckland, Auckland, New Zealand, in 2014. He is currently a Scientist with the Singapore Institute of Manufacturing Technology, Agency for Science, Technology and Research, Singapore. He also holds a joint appointment with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He has diverse research experience in computational science. His current main interests lie in the theory and algorithms of transfer and multitask optimization, neuroevolution, surrogate modeling, and scientific machine learning.

Dr. Abhishek is the recipient of the 2019 and the 2023 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, for foundational works on evolutionary multitasking. He received the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE 2021 Outstanding Associate Editor Award. He is also an Editorial Board Member of the Complex & Intelligent Systems, the Memetic Computing, and Adaptation, Learning, and Optimization (Springer).

Zefeng Chen received the B.Sc. degree in information and computational science from Sun Yat-sen University (SYSU), Guangzhou, China, in 2013, and the M.Sc. degree in computer science and technology from the South China University of Technology, Guangzhou, China, in 2016, and the Ph.D. degree in computer science and Technology from SYSU in 2019.

He was a Postdoctoral Research Fellow working with Prof. Y.-S. Ong with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, from October 2019 to October 2021. He is currently an Assistant Professor with the School of Artificial Intelligence, SYSU. His current research interests mainly include evolutionary computation, evolutionary learning, and data-driven optimization.

Yew-Soon Ong (Fellow, IEEE) received the Ph.D. degree in artificial intelligence in complex engineering design from the University of Southampton, Southampton, U.K., in 2003.

He is a President Chair Professor of Computer Science with the Nanyang Technological University (NTU), Singapore, and concurrently the Chief Artificial Intelligence Scientist of the Agency for Science, Technology and Research, Singapore. At NTU, he serves as the Co-Director of the Singtel-NTU Cognitive & Artificial Intelligence Joint Laboratory. His core research interest is in artificial and computational intelligence, where he has received four IEEE outstanding paper awards.

Dr. Ong was listed as a Thomson Reuters Highly Cited Researcher and among the World’s Most Influential Scientific Minds. He is the inaugural Editor-in-Chief of the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE and an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, and IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE.