From Multi-Task Gradient Descent to Gradient-Free Evolutionary Multitasking: A Proof of Faster Convergence

Lu Bai, Wu Lin, Abhishek Gupta, and Yew-Soon Ong, *Fellow, IEEE*

***Abstract*—Evolutionary multitasking which solves multiple optimization tasks simultaneously has gained increasing research attention in recent years. By utilizing the useful information from related tasks while solving the tasks concurrently, improved performance has been shown in various problems. Despite the success enjoyed by the existing evolutionary multitasking algorithms, there still is a lack of theoretical studies guaranteeing faster convergence compared to the conventional single task case. To analyze the effects of transferred information from related tasks, in this paper, we first put forward a novel multi-task gradient descent (MTGD) algorithm, which enhances the standard gradient descent updates with a multi-task interaction term. The convergence of the resultant MTGD is derived, furthermore, we present the first proof of faster convergence of MTGD relative to its single task counterpart. Utilizing MTGD, we formulate a gradient-free evolutionary multitasking algorithm named as multi-task evolution strategies (MTES). Importantly, the single task evolution strategies (ES) we utilize is shown to asymptotically approximate gradient descent, and hence the faster convergence results derived for MTGD extend to the case of MTES as well. Numerical experiments comparing MTES with single task ES on synthetic benchmarks and practical optimization examples serve to substantiate our theoretical claim.**

***Index Terms*—Multi-Task Optimization, Gradient Descent, Faster Convergence, Evolution Strategies**

I. Introduction

Solving optimization problems is ubiquitous in various areas, such as mechanics, economics, operation management, and machine learning [1], [2], [3], [4]. Traditionally, optimization problems are usually solved independently. However, many real-world problems have similarities in nature, solving one problem may provide useful information for other similar problems [5], [6], [7]. Multi-task optimization allows us to leverage inter-task similarities to *reproduce* high-quality performance across related tasks [8], similar to what humans are inherently capable of.

To perform multi-task optimization, evolutionary multitasking has been proposed to solve multiple different but related optimization problems simultaneously in recent years. In [9], a multifactorial evolutionary algorithm (MFEA) is proposed to solve complementary tasks in a unified search space, where the useful information is transferred among different tasks through the chromosomal crossover during the evolution. To allay the susceptibility to harmful interactions when the similarity between the tasks is low, [10] proposes the MFEA-II where the similarities are explored adaptively and locally for different task pairs. Realizations of the evolutionary multitasking engine within the domain of multi-objective optimization are given in [11], [12]. Based on MFEA, two improvements include detecting the occurrence of parting ways, at when the information sharing begins to fail, and reallocating resources on fitness evaluations are made in [13] to improve solution quality. [14] introduces MFEA with a representation scheme based on the Cayley Code to utilize the strengths of MFEA solving the clustered tree problems. To enable multiple biases embedded in different evolutionary search operators, [15] and [16] propose evolutionary multitasking paradigms with multi-populations and explicit genetic transfer across tasks to promote information transfer. From examples such as the above, a common thread we find in the majority of existing multitasking algorithms is that the search has so far been limited to occur in a gradient-free manner.

The effectiveness of evolutionary multitasking has been verified on a range of practical optimization problems. Such as solving multiple permutation-based optimization problems (e.g., travel salesman problem, job-shop scheduling problem, and capacitated vehicle routing problem) [17], [18], generalized vehicle routing problem with occasional drivers [19], clustered shortest path tree problem decoupled as multiple tasks [20], evolutionary neural networks for multi-step chaotic time series problems [21], and complex simulation-based engineering design [22]. Despite these promising empirical results, there remains a gaping lack of theoretical analysis of how the transferred information from related tasks enhances the optimization process. Even though the recent MFEA-II paper [10] presents a proof of convergence and analyzes the effects of inter-task interactions, it does not offer proof of faster convergence. To the best of our knowledge, no theoretical guarantee of faster convergence via multitasking has been

Manuscript received May 6, 2020; revised October 24, 2020; accepted January 3, 2021. This work was supported in part by the A*STAR Cyber-Physical Production System (CPPS) – Towards Contextual and Intelligent Response Research Program, under the RIE2020 IAF-PP Grant A19C1a0018, and in part by the Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU), which is a collaboration between Singapore Telecommunications Limited (Singtel) and Nanyang Technological University (NTU) that is funded by the Singapore Government through the Industry Alignment Fund - Industry Collaboration Projects Grant.

Lu Bai and Yew-Soon Ong are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: bailu@ntu.edu.sg; asysong@ntu.edu.sg).

Wu Lin is with the College of Computer Science and Software Engineering, Shenzhen University, China (e-mail: linwu2017@email.szu.edu.cn).

Abhishek Gupta is with the Singapore Institute of Manufacturing Technology, Agency for Science, Technology and Research, Singapore (e-mail: abhishek_gupta@simtech.a-star.edu.sg).

obtained.

For an optimization problem, one basic idea is to utilize the gradient of a differentiable objective function as the searching direction, which will lead to better solutions as long as the update steps are sufficiently small. In virtue of the effectiveness of gradient-based iteration, in this paper, we first propose a novel multi-task gradient descent (MTGD) framework to solve multiple minimization problems simultaneously. In the MTGD, the decision variable of each problem is updated based on the gradient descent and information transferred from related tasks during the iteration. The transfer intensity is determined adaptively through the transfer coefficients, which are designed to satisfy certain conditions to maintain convergence behavior. By adopting the properties of gradient descent iteration, the convergence of MTGD is theoretically analyzed. Furthermore, for problems that satisfy certain conditions, faster convergence of MTGD over single task gradient descent is proven for the first time.

Though gradient descent is effective in solving optimization problems, in many practical problems, the precise analytic form of the objective function is unknown or non-differentiable, which leads to no gradient information, or the gradient can not be computed directly or efficiently. Gradient-free optimization shows its strength in solving this kind of problems since it only relies on fitness evaluations at certain sample points. The state-of-the-art OpenAI ES [23] is a recently proposed algorithm that constitutes a more principled approach to black-box optimization. By updating the distribution from which the candidate solutions are drawn using approximated gradients (with respect to the distribution parameters), better solutions are more likely to be sampled than typical randomized evolutionary operators. By combining the gradient approximation in OpenAI ES with the proposed MTGD, we develop a gradient-free multitasking algorithm named as multi-task evolution strategies (MTES). In the MTES, OpenAI ES is employed as the base solver for each task. Importantly, the update in the OpenAI ES is shown to asymptotically approximate gradient descent, and hence the results derived for MTGD are in principle extendible to the gradient-free MTES as well. In other words, a proof of faster convergence for gradient-free multitasking is effectively obtained in the limiting case. This theoretical claim will be empirically verified in the paper via synthetic benchmarks and reinforcement learning examples.

To summarize, in this paper, we first propose a novel MTGD framework, and then develop an associated gradient-free MTES, to solve multiple minimization problems simultaneously. In comparison to existing works, our main contributions are threefold:

1) We propose the MTGD framework and provide conditions on the transfer coefficients for convergence. It is worth noting that any transfer mechanisms that satisfy the given conditions will not impede the overall convergence behavior, which in turn offers the flexibility in designing different transfer mechanisms for different purposes.
2) The convergence of MTGD is theoretically analyzed, and furthermore, for problems that satisfy certain conditions, we present the first proof of faster convergence of MTGD relative to its single task counterpart.
3) A novel MTES utilizing the OpenAI ES is developed based on the MTGD, thus inheriting its convergence results in the limiting case. Experiments using MTES for solving synthetic multi-task benchmark functions, practical double-pole balancing problems, and parameterized planar arm problems validate the proposed algorithm.

The rest of the paper is organized as follows. Section II gives preliminaries including some basics of gradient descent iteration and the OpenAI ES. Then, the proposed MTGD is presented in Section III, where the convergence of MTGD is analyzed and we prove that the convergence of MTGD is faster than single task gradient descent under certain conditions. In Section IV, the proposed MTES is detailed and one scheme for calculating the transfer coefficients between different tasks is presented. Experiments on the synthetic multi-task benchmark functions, practical double-pole balancing problems, and parameterized planar arm problems are conducted in Section V to evaluate the performance. Finally, Section VI gives the conclusion.

II. Preliminary

In this section, we introduce some basic results of the single task gradient descent iteration and the OpenAI ES which is used as the base solver in MTES.

A. Gradient Descent Iteration

Suppose the cost function $f(\theta) : \mathbb{R}^d \to \mathbb{R}$ is strongly convex, twice differentiable, and the gradient of f is Lipschitz continuous with constant L_f, i.e.,

$$\|\nabla f(\theta_1) - \nabla f(\theta_2)\| \leq L_f \|\theta_1 - \theta_2\|, \ \forall \theta_1, \theta_2 \in \mathbb{R}^d.$$

Since f is strongly convex and twice differentiable, there exists positive constant ξ such that

$$\nabla^2 f(\theta) \geq \xi I_d,$$

where I_d is the identity matrix of dimension d. As a result, we have the following relation,

$$\xi I_d \leq \nabla^2 f(\theta) \leq L_f I_d, \ \forall \theta,$$

where $\xi \leq L_f$.

The gradient descent updates θ as follows

$$\theta^{t+1} = \theta^t - \alpha \nabla f(\theta^t), \tag{1}$$

where α is the step size satisfying $\alpha \leq \frac{1}{L_f}$. By using the gradient descent iteration, θ asymptotically converges to the optimal solution at which f takes the minimum.

B. Evolution Strategies

In the OpenAI ES, the candidate solution is drawn from an isotropic multivariate Gaussian distribution $\mathcal{N}(\theta, \sigma^2 I_d)$, where θ is the d-dimensional mean vector and $\sigma^2 I_d$ is the covariance. The mean vector θ of the population is updated and the standard deviation σ is fixed during the evolution. Denote the objective function as $F : \mathbb{R}^d \to \mathbb{R}$. The aim is to find the minimum of F. The procedure of the OpenAI ES is summarized in Algorithm 1 [23]. In the rest of this paper, ES refers to Algorithm 1.

Algorithm 1 OpenAI Evolution Strategies [23]

Input: Objective function F, standard deviation σ, initial parameter θ^0, learning rate α

Set $t = 0$
while stopping criterion is not met **do**
  for $k = 1, ..., n$ **do**
    Draw sample $\epsilon_k \sim \mathcal{N}(0, I_d)$
    Calculate fitness $F_k = F(\theta^t + \sigma\epsilon_k)$
  end for
  $\theta^{t+1} = \theta^t - \alpha \frac{1}{n\sigma} \sum_{k=1}^{n} F_k \epsilon_k$
  $t = t + 1$
end while

III. Gradient-Based Multi-Task Iteration

In this section, we derive and present the multi-task gradient descent (MTGD) algorithm, which solves multiple minimization tasks simultaneously based on gradient descent and information from other tasks. In particular, we first present the overall expression of the MTGD and conditions on the transfer coefficients. Followed is the stability analysis of the MTGD, which shows that while incorporating information from other tasks, the iteration still converges to its original optimal solution under the given transfer coefficients. Then, the comparison of the convergence rate between MTGD and the usual single task gradient descent is made, where we show that under certain conditions, the convergence of MTGD is faster than the single task gradient descent.

A. Notations

Throughout this paper, I_n denotes identity matrix of size $n \times n$, $\otimes$ denotes the Kronecker product. $[x_i]_{\text{vec}}$ denotes a column vector formed by stacking x_i on top of each other, i.e., $[x_i]_{\text{vec}} = [x_1', ..., x_T']'$, and diag$\{x_i\}$ denotes a diagonal matrix with the i-th diagonal element being x_i. The norm $\|\cdot\|$ without specifying the subscript represents the Euclidean norm by default.

Suppose we have T tasks to be solved. Each task $i \in \{1, ..., T\}$ aims to solve the following optimization problem,

$$\min_{\theta_i} \ f_i(\theta_i), \tag{2}$$

where $\theta_i \in \mathbb{R}^d$ is the decision variable of task i, which is of dimension d. To solve the optimization problem, gradient of the objective function, which is ∇f_i, is utilized as the searching direction. Furthermore, to utilize information from similar tasks, we use transfer coefficient m_{ij} to determine the information flow from task j to task i. The notations used in the forthcoming proposed MTGD is summarized in TABLE I.

TABLE I: Notation used by MTGD

Notation	Meaning
T	Number of tasks
θ_i	Decision variable of task i
f_i	Objective function of task i
∇f_i	Gradient of f_i
α	Step size
m_{ij}	Transfer coefficient describes the information flow from task j to task i

B. Multi-Task Gradient Descent

Based on the assumption that there are similarities among the tasks, it is believed that the solution of one task may facilitate the search for the solution of another task. To utilize the similarities among the tasks, the values of the decision variables during the iteration process are transferred between the tasks. Following this intuition, we propose the MTGD iteration as follows,

$$\theta_i^{t+1} = \sum_{j=1}^{T} m_{ij}^t \theta_j^t - \alpha \nabla f_i(\theta_i^t), \tag{3}$$

where m_{ij}^t is the transfer coefficient at iteration t. If there is information transfer from task j to task i at iteration t, $m_{ij}^t > 0$, otherwise, $m_{ij}^t = 0$. The transfer coefficient m_{ij}^t is required to satisfy the following conditions,

$$m_{ij}^t \geq 0, \tag{4a}$$

$$\sum_{j=1}^{T} m_{ij}^t = 1, \tag{4b}$$

$$m_{ij}^t = 0, \text{ for } t > T_0 \text{ and } j \neq i, \tag{4c}$$

where T_0 is a nonnegative integer. (4a) implies that inter-task interactions are non-repulsive, (4b) imposes a sum-to-one normalization on the transfer coefficient, which is used to limit the transfer power to prevent divergence, and (4c) implies no transfer occurs after time T_0.

C. Convergence Analysis

In this section, we give the convergence property of the proposed MTGD. As can be seen from (4c), MTGD will degrade into single task gradient descent after T_0. Although the convergence can be ensured by the single task gradient descent, the incorporation of information from other tasks before T_0 may make the iteration diverge without proper transfer mechanism, which will make the single task gradient descent after T_0 take a much longer time to converge. The result in this section ensures that the proposed MTGD can obtain a bounded value at T_0, which serves as the basis of obtaining a faster convergence compared to the single task gradient descent.

From (4b), we have $m_{ii}^t = 1 - \sum_{j \neq i} m_{ij}^t$. Rewrite iteration (3) as follows

$$\begin{aligned}\theta_i^{t+1} =& m_{ii}^t \theta_i^t + \sum_{j \neq i} m_{ij}^t \theta_j^t - \alpha \nabla f_i(\theta_i^t) \\ =& (1 - \sum_{j \neq i} m_{ij}^t)\theta_i^t + \sum_{j \neq i} m_{ij}^t \theta_j^t - \alpha \nabla f_i(\theta_i^t).\end{aligned} \tag{5}$$

Rescale m_{ij}^t as

$$\bar{m}_{ij}^t = \begin{cases} \frac{1}{\alpha\sigma} m_{ij}^t, & j \neq i, \\ 1 - \frac{1}{\alpha\sigma} \sum_{j \neq i} m_{ij}^t, & j = i, \end{cases}$$

where σ is a positive constant and satisfies the condition

$$1-\frac{1}{\alpha\sigma}\sum_{j\neq i} m_{ij}^t > 0.$$

The rescaling keeps $\sum_{j=1}^T \bar{m}_{ij}^t = 1$ and the parameter $\alpha\sigma$ rescales the relative importance of own information and information from related tasks. With the rescaling, the iteration (5) can be expressed as

$$\begin{aligned}\theta_i^{t+1} =&(1-\alpha\sigma\sum_{j\neq i}\bar{m}_{ij}^t)\theta_i^t+\alpha\sigma\sum_{j\neq i}\bar{m}_{ij}^t\theta_j^t-\alpha\nabla f_i(\theta_i^t)\\ =&\theta_i^t-\alpha\sigma(1-\bar{m}_{ii}^t)\theta_i^t+\alpha\sigma\sum_{j\neq i}\bar{m}_{ij}^t\theta_j^t-\alpha\nabla f_i(\theta_i^t)\\ =&(1-\alpha\sigma)\theta_i^t+\alpha\sigma\sum_{j=1}^T\bar{m}_{ij}^t\theta_j^t-\alpha\nabla f_i(\theta_i^t). \end{aligned} \tag{6}$$

Let the i,j-th element of $\bar{M}^t\in\mathbb{R}^{T\times T}$ at iteration time t being $\bar{m}_{ij}^t$, denote $\bar{\mathcal{M}}^t=\bar{M}^t\otimes I_d\in\mathbb{R}^{dT\times dT}$, $\Theta=[\theta_1',...,\theta_T']'\in\mathbb{R}^{dT}$, and $\nabla f(\Theta^t)=[\nabla f_1(\theta_1^t)',...,\nabla f_T(\theta_T^t)']'\in\mathbb{R}^{dT}$. Write (6) into a concatenated form gives

$$\Theta^{t+1}=(1-\alpha\sigma)\Theta^t+\alpha\sigma\bar{\mathcal{M}}^t\Theta^t-\alpha\nabla f(\Theta^t). \tag{7}$$

Denote $\tilde{\theta}_i^{\,t}=\theta_i^t-\theta_i^*$, where θ_i^* is the optimal solution for task i, $\tilde{\Theta}^t=[\tilde{\theta}_1^{t'},...,\tilde{\theta}_T^{t'}]'$, $\Theta^*=[\theta_1^{*'},...,\theta_T^{*'}]'$, $\bar{L}_{f_i}=\max_i\{L_{f_i}\}$, $\underline{\xi_i}=\min_i\{\xi_i\}$, and $\|\cdot\|$ denotes the Euclidean norm. In the following theorem, we show that for a small step-size α, the error vector $\tilde{\theta}_i^t$ converges to zero as $t\to\infty$ under the MTGD (7). We use the block maximum norm defined in [24] to show the convergence of the above iteration. $\|\cdot\|$ represents the Euclidean norm. The block maximum norm of a vector $x=[x_i]_{\text{vec}}\in\mathbb{R}^{dT}$ with $x_i\in\mathbb{R}^d$ is defined as [24]

$$\|x\|_{b,\infty}=\max_i\|x_i\|.$$

The induced matrix block maximum norm for a matrix $A\in\mathbb{R}^{n\times d}$ is therefore defined as [24]

$$\|A\|_{b,\infty}=\max_{x\neq 0}\frac{\|Ax\|_{b,\infty}}{\|x\|_{b,\infty}}.$$

Theorem 1. *Under the MTGD* (7) *with the transfer coefficient* $\bar{m}_{ij}^t$ *satisfies*

$$\begin{aligned}&\sum_{j=1}^T\bar{m}_{ij}^t=1,\\ &\bar{m}_{ij}^t\geq 0,\\ &\bar{m}_{ij}^t=0,\ \textit{for } t>T_0 \textit{ and } j\neq i,\end{aligned}$$

and the step size α *satisfies*

$$0<\alpha<\frac{2}{2\sigma+\bar{L}_{f_i}}, \tag{8}$$

we have

$$\begin{aligned}\|\tilde{\Theta}^{T_0}\|_{b,\infty}\leq&\beta_1^{T_0}\|\tilde{\Theta}^0\|_{b,\infty}+\alpha\sigma b_0\frac{1-\beta_1^{T_0}}{1+\beta_1},\\ \|\tilde{\Theta}^t\|_{b,\infty}\leq&\beta_2^{t-T_0}\|\tilde{\Theta}^{T_0}\|_{b,\infty},\ t>T_0,\end{aligned}$$

where $\beta_1=\max_i\{|1-\alpha\sigma-\alpha\xi_i|,|1-\alpha\sigma-\alpha L_{f_i}|\}+\alpha\sigma$, $\beta_2=\max_i\{|1-\alpha\xi_i|,|1-\alpha L_{f_i}|\}$, *and* $b_0=\max_{i,j}\|\theta_i^*-\theta_j^*\|$.

Proof. Subtracting Θ^* from both sides of (7) gives

$$\begin{aligned}\tilde{\Theta}^{t+1}=&((1-\alpha\sigma)I_{dT}+\alpha\sigma\bar{\mathcal{M}}^t)\Theta^t-\Theta^*-\alpha\nabla f(\Theta^t)\\ =&((1-\alpha\sigma)I_{dT}+\alpha\sigma\bar{\mathcal{M}}^t-\alpha H^t)\tilde{\Theta}^t\\ &+\alpha\sigma(\bar{\mathcal{M}}^t-I_{dT})\Theta^*,\end{aligned} \tag{9}$$

where $H^t=\int_0^1\nabla^2 f(\Theta^*+\mu(\Theta^t-\Theta^*))d\mu\in\mathbb{R}^{dT\times dT}$. It can be verified that H^t is a block diagonal matrix and the block diagonal elements $H_i^t=\int_0^1\nabla^2 f_i(\theta_i^*+\mu(\theta_i^t-\theta_i^*))d\mu\in\mathbb{R}^{d\times d}$ for $i=1,...,T$ are Hermitian. From the iteration in (9) we have

$$\begin{aligned}&\|\tilde{\Theta}^{t+1}\|_{b,\infty}\leq\|((1-\alpha\sigma)I_{dT}+\alpha\sigma\bar{\mathcal{M}}^t-\alpha H^t)\tilde{\Theta}^t\|_{b,\infty}\\ &+\alpha\sigma\|(\bar{\mathcal{M}}^t-I_{dT})\Theta^*\|_{b,\infty}\\ \leq&\|(1-\alpha\sigma)I_{dT}+\alpha\sigma\bar{\mathcal{M}}^t-\alpha H^t\|_{b,\infty}\|\tilde{\Theta}^t\|_{b,\infty}\\ &+\alpha\sigma\|(\bar{\mathcal{M}}^t-I_{dT})\Theta^*\|_{b,\infty}\\ \leq&(\|(1-\alpha\sigma)I_{dT}-\alpha H^t\|_{b,\infty}+\alpha\sigma\|\bar{\mathcal{M}}^t\|_{b,\infty})\|\tilde{\Theta}^t\|_{b,\infty}\\ &+\alpha\sigma\|(\bar{\mathcal{M}}^t-I_{dT})\Theta^*\|_{b,\infty}.\end{aligned}$$

From Lemma D.3 in [24], we have

$$\|\bar{\mathcal{M}}^t\|_{b,\infty}=\|\bar{M}^t\|_\infty=1,$$

where the last equality comes from the fact that $\bar{m}_{ij}^t\geq 0$ and the row summation of $\bar{M}^t$ is one. Since $\xi_i I_d\leq\nabla^2 f_i(\theta_i)\leq L_{f_i}I_d$, $\xi_i I_d\leq\int_0^1\nabla^2 f_i(\theta_i^*+\mu(\theta_i-\theta_i^*))d\mu\leq L_{f_i}I_d$. Thus, $\|(1-\alpha\sigma)I_d-\alpha H_i^t\|\leq\gamma_i$ where $\gamma_i=\max\{|1-\alpha\sigma-\alpha\xi_i|,|1-\alpha\sigma-\alpha L_{f_i}|\}$. By the definition of induced matrix block maximum norm, we have

$$\begin{aligned}&\|(1-\alpha\sigma)I_{dT}-\alpha H^t\|_{b,\infty}\\ =&\max_{\boldsymbol{x}\neq 0}\frac{\|((1-\alpha\sigma)I_{dT}-\alpha H^t)\boldsymbol{x}\|_{b,\infty}}{\|\boldsymbol{x}\|_{b,\infty}}\\ =&\max_{\boldsymbol{x}\neq 0}\frac{\max_i\|((1-\alpha\sigma)I_d-\alpha H_i^t)\boldsymbol{x}_i\|}{\|\boldsymbol{x}\|_{b,\infty}}\\ \leq&\max_{\boldsymbol{x}\neq 0}\frac{\max_i\|(1-\alpha\sigma)I_d-\alpha H_i^t\|\|\boldsymbol{x}\|_{b,\infty}}{\|\boldsymbol{x}\|_{b,\infty}}\\ =&\max_i\|(1-\alpha\sigma)I_d-\alpha H_i^t\|\\ \leq&\bar{\gamma},\end{aligned}$$

where $\bar{\gamma}=\max\{\gamma_i\}$. Thus,

$$\|\tilde{\Theta}^{t+1}\|_{b,\infty}\leq(\bar{\gamma}+\alpha\sigma)\|\tilde{\Theta}^t\|_{b,\infty}+\alpha\sigma\|(\bar{\mathcal{M}}^t-I_{dT})\Theta^*\|_{b,\infty}. \tag{10}$$

By choosing the step size α to satisfy $\bar{\gamma}+\alpha\sigma<1$, the iteration asymptotically converges. To ensure $\bar{\gamma}+\alpha\sigma<1$, it is sufficient to ensure

$$|1-\alpha\sigma-\alpha\xi_i|+\alpha\sigma<1 \text{ and } |1-\alpha\sigma-\alpha L_{f_i}|+\alpha\sigma<1,\ \forall i,$$

which leads to

$$0<\alpha<\frac{2}{2\sigma+\bar{L}_{f_i}}.$$

Before T_0, since $\bar{M}^t$ is row-sum-to-one and the elements of $\bar{M}^t$ are all non-negative, $\bar{\mathcal{M}}^t\Theta^*$ is a convex combination of $\theta_i^*, i\in\{1,...,T\}$. Thus, $\|(\bar{\mathcal{M}}^t-I_{dT})\Theta^*\|_{b,\infty}$ is upper

bounded by $\max_{i,j}\|\theta_i^*-\theta_j^*\|$. Denote $b_0=\max_{i,j}\|\theta_i^*-\theta_j^*\|$ and $\beta_1=\bar{\gamma}+\alpha\sigma<1$. From the iteration in (10), we have

$$\begin{aligned}\|\tilde{\Theta}^{T_0}\|_{b,\infty}\leq&\beta_1\|\tilde{\Theta}^{T_0-1}\|_{b,\infty}+\alpha\sigma b_0\\ \leq&\cdots\\ \leq&\beta_1^{T_0}\|\tilde{\Theta}^0\|_{b,\infty}+\alpha\sigma b_0\sum_{k=0}^{T_0-1}\beta_1^k\\ =&\beta_1^{T_0}\|\tilde{\Theta}^0\|_{b,\infty}+\alpha\sigma b_0\frac{1-\beta_1^{T_0}}{1+\beta_1}\end{aligned}\tag{11}$$

After T_0, since $\bar{M}^t=I_T$, the iteration (9) becomes

$$\tilde{\Theta}^{t+1}=(I_{dT}-\alpha H^t)\tilde{\Theta}^t.$$

Denote $\beta_2=\max_i\{|1-\alpha\xi_i|,|1-\alpha L_{f_i}|\}$. Under the condition (8), $\beta_2<1$. Then, we have for $t>T_0$

$$\begin{aligned}\|\tilde{\Theta}^t\|_{b,\infty}\leq&\beta_2\|\tilde{\Theta}^{t-1}\|_{b,\infty}\\ \leq&\beta_2^{t-T_0}\|\tilde{\Theta}^{T_0}\|_{b,\infty}.\end{aligned}$$

Substituting (11) into the above inequality gives

$$\|\tilde{\Theta}^t\|_{b,\infty}\leq\beta_2^{t-T_0}\beta_1^{T_0}\|\tilde{\Theta}^0\|_{b,\infty}+\alpha\sigma b_0\frac{(1-\beta_1^{T_0})\beta_2^{t-T_0}}{1+\beta_1}.$$

As a result, we have

$$\lim_{t\to\infty}\|\tilde{\Theta}^t\|_{b,\infty}\to 0.$$

From the definition of block maximum norm, $\lim_{t\to\infty}\|\tilde{\theta}_i^t\|=0$ is obtained. □

D. Faster Convergence under Symmetric Transfer Matrix

In this section, with additional assumption that the transfer coefficient satisfies $m_{ij}^t=m_{ji}^t$, we compare the convergence rate of the MTGD (3) and the single task gradient descent (1), and give a sufficient condition that the convergence rate of (3) is faster than (1).

A graph $\mathcal{G}^t=\{\mathcal{V},\mathcal{E}^t\}$ is used to describing the information transfer at iteration t, where $\mathcal{V}=\{1,...,T\}$ is the task set and $\mathcal{E}^t\subseteq\mathcal{V}\times\mathcal{V}$ is the edge set. An edge $(i,j)\in\mathcal{E}^t$ means there is information transfer between task i and task j. Under the assumption that $m_{ij}^t=m_{ji}^t$, the graph $\mathcal{G}^t$ is undirected. A graph $\mathcal{G}^t$ corresponds to a matrix M^t. When the graph is connected, M^t has one eigenvalue 1 with corresponding eigenvector $\mathbf{1}_T$, where $\mathbf{1}_T$ represents a T dimensional column vector with each element being 1. When there are p isolated subgraphs with each subgraph connected, there are p eigenvalues 1. When there are p isolated subgraphs, it means that there are p groups of related tasks. Under this condition, each group of related tasks can be handled separately. Thus, without loss of generality, we assume that the graph $\mathcal{G}^t$ is connected.

Suppose M^t is chosen from a finite set $\mathbb{M}=\{M_1,...,M_m\}$. Denote $\eta_1=\max(\rho(\mathcal{M}^t-\alpha H^t))$ and $\eta_2=\max(\rho(I_{dT}-\alpha H^t))$, where $\rho(\cdot)$ represents the spectral radius of a matrix. We state in the following theorem that under certain conditions, MTGD is faster than single task gradient descent.

Theorem 2. *Suppose there exist i and j such that $H_i^t\neq H_j^t$ and the transfer coefficient m_{ij}^t satisfies*

$$\begin{aligned}&m_{ij}^t\geq 0,\\ &m_{ii}^t\geq 0.5,\\ &m_{ij}^t=m_{ji}^t,\\ &\sum_{j=1}^T m_{ij}^t=1,\\ &m_{ij}^t=0\ for\ t>T_0\ and\ j\neq i,\end{aligned}$$

where T_0 is a nonnegative integer satisfying

$$\eta_1^{T_0}\frac{\|\nabla f(\Theta^0)\|}{\bar{L}_{f_i}}+\frac{1-\eta_1^{T_0}}{1+\eta_1}b_0<\eta_2^{T_0}\frac{\|\nabla f(\Theta^0)\|}{\bar{L}_{f_i}},\tag{12}$$

then, under the MTGD (3)*, $\|\tilde{\Theta}^t\|$ converges to zero faster when $T_0>0$ than under the single task gradient descent* (1) *if the step size α satisfies*

$$0<\alpha<\frac{1}{2\bar{L}_{f_i}}.\tag{13}$$

Proof. Write (3) into a concatenated form gives

$$\Theta^{t+1}=\mathcal{M}^t\Theta^t-\alpha\nabla f(\Theta^t).$$

Subtracting Θ^* from both sides of the above equation, we have

$$\begin{aligned}\tilde{\Theta}^{t+1}=&\mathcal{M}^t\tilde{\Theta}^t+(\mathcal{M}^t-I_{dT})\Theta^*-\alpha(\nabla f(\Theta^t)-\nabla f(\Theta^*))\\ =&(\mathcal{M}^t-\alpha H^t)\tilde{\Theta}^t+(\mathcal{M}^t-I_{dT})\Theta^*\\ =&A_m^t\tilde{\Theta}^t+(\mathcal{M}^t-I_{dT})\Theta^*,\end{aligned}$$

where $A_m^t=\mathcal{M}^t-\alpha H^t$. Due to the assumption that M^t is symmetric, A_m^t is symmetric. Thus,

$$\begin{aligned}\|\tilde{\Theta}^{t+1}\|\leq&\|A_m^t\tilde{\Theta}^t\|+\|(\mathcal{M}^t-I_{dT})\Theta^*\|\\ \leq&\rho(A_m^t)\|\tilde{\Theta}^t\|+\|(\mathcal{M}^t-I_{dT})\Theta^*\|\\ \leq&\prod_{s=0}^t\rho(A_m^s)\|\tilde{\Theta}^0\|+\|(\mathcal{M}^t-I_{dT})\Theta^*\|\\ &+\sum_{s=0}^{t-1}\prod_{r=s+1}^t\rho(A_m^r)\|(\mathcal{M}^s-I_{dT})\Theta^*\|.\end{aligned}\tag{14}$$

Under single task gradient descent (1), the iteration of Θ_s is

$$\begin{aligned}\tilde{\Theta}_s^{t+1}=&\Theta_s^t-\Theta^*-\alpha\nabla f(\Theta_s^t)\\ =&(I_{dT}-\alpha H^t)\tilde{\Theta}_s^t.\end{aligned}$$

Thus,

$$\|\tilde{\Theta}_s^{t+1}\|\leq\prod_{s=0}^t\rho(A_s^s)\|\tilde{\Theta}^0\|,\tag{15}$$

where $A_s^t=I_{dT}-\alpha H^t$.

Let $\Delta^t=I_{dT}-\mathcal{M}^t$, it is obvious that

$$A_s^t=A_m^t+\Delta^t.\tag{16}$$

Let $\lambda_m(B)$ being the decreasing ordered eigenvalues of matrix $B\in\mathbb{R}^{dT\times dT}$, i.e., $\lambda_1(B)\geq\lambda_2(B)\geq\cdots\geq\lambda_{dT}(B)$. Let $\boldsymbol{r}_i$ be the right eigenvector of $\mathcal{M}^t$ corresponding to eigenvalue

$\lambda_i(\mathcal{M}^t)$, i.e., $\mathcal{M}^t\boldsymbol{r}_i = \lambda_i(\mathcal{M}^t)\boldsymbol{r}_i$. Then, $\boldsymbol{r}_i$ is also a right eigenvector of Δ^t corresponding to eigenvalue $1-\lambda_i(\mathcal{M}^t)$, which can be obtained by the following relationship,

$$\Delta^t\boldsymbol{r}_i = (I_{dT}-\mathcal{M}^t)\boldsymbol{r}_i = \boldsymbol{r}_i - \lambda_i(\mathcal{M}^t)\boldsymbol{r}_i = (1-\lambda_i(\mathcal{M}^t))\boldsymbol{r}_i.$$

Thus, the eigenvalues of matrix Δ^t is $\lambda_1(\Delta^t) = 1-\lambda_{dT}(\mathcal{M}^t)$ and $\lambda_{dT}(\Delta^t) = 1-\lambda_1(\mathcal{M}^t) = 0$. As a result, Δ^t is positive semidefinite. Since A_s^t, A_m^t, Δ^t are all Hermitian matrix and $\Delta^t \geq 0$, from Weyl's theorem [25], we have

$$\lambda_i(A_m^t) \leq \lambda_i(A_s^t), i = 1, ..., dT.$$

The equality holds if and only if there is a nonzero vector $\boldsymbol{x}$ such that $A_m^t\boldsymbol{x} = \lambda_i(A_m^t)\boldsymbol{x}$, $\Delta^t\boldsymbol{x} = 0$, and $A_s^t\boldsymbol{x} = \lambda_i(A_s^t)\boldsymbol{x}$. For $\Delta^t\boldsymbol{x} = 0$, we have $\boldsymbol{x} = \mathcal{M}^t\boldsymbol{x}$, which indicates that $\boldsymbol{x}$ is in the space spanned by the columns of $\mathbf{1}_T \otimes I_d$ for $M^t \neq I_T$. $A_s^t\boldsymbol{x} = \lambda_1(A_s^t)\boldsymbol{x}$ indicates $\boldsymbol{x}-\alpha H^t\boldsymbol{x} = \lambda_1(A_s^t)\boldsymbol{x}$, which gives $\alpha H^t\boldsymbol{x} = (1-\lambda_1(A_s^t))\boldsymbol{x}$. Since there exist i, j such that $H_i^t \neq H_j^t$, such $\boldsymbol{x}$ does not exist. As a result, $\lambda_1(A_m^t) < \lambda_1(A_s^t)$.

From the relation $A_m^t = \mathcal{M}^t - \alpha H^t$ and the Weyl's theorem, we have

$$\lambda_i(\mathcal{M}^t) + \lambda_{dT}(-\alpha H^t) \leq \lambda_i(A_m^t) \leq \lambda_i(\mathcal{M}^t) + \lambda_1(-\alpha H^t).$$

Since $\xi_i I_d \leq \nabla^2 f_i(\theta_i) \leq L_{f_i} I_d$, $\xi_i \leq \lambda_i(H^t) \leq \bar{L}_{f_i}$. From the stochastic property of M^t, $\lambda_1(\mathcal{M}^t) = \lambda_1(M^t) = 1$ and $\lambda_{dT}(\mathcal{M}^t) = \lambda_T(M^t) \geq -1$. Thus

$$\begin{aligned} 1-\alpha\bar{L}_{f_i} \leq &\lambda_1(A_m^t) \leq 1-\alpha\underline{\xi_i}, \\ \lambda_T(M^t)-\alpha\bar{L}_{f_i} \leq &\lambda_{dT}(A_m^t) \leq \lambda_T(M^t)-\alpha\underline{\xi_i}. \end{aligned}$$

Under the following conditions

$$\begin{aligned} \alpha &< \frac{1}{2\bar{L}_{f_i}}, \\ m_{ii}^t &\geq 0.5, \end{aligned}$$

we have $2\alpha\bar{L}_{f_i} - 1 < 0$ and $\lambda_T(M^t) \geq 0$. Thus,

$$|\lambda_T(M^t) - \alpha\bar{L}_{f_i}| < 1-\alpha\bar{L}_{f_i}.$$

As a result, $\rho(A_m^t) = \max\{|\lambda_1(A_m^t)|, |\lambda_{dT}(A_m^t)|\} = \lambda_1(A_m^t)$. Together with the result that $\lambda_1(A_m^t) < \lambda_1(A_s^t)$, we can conclude that $\rho(A_m^t) < \rho(A_s^t)$ for $M^t \neq I_T$.

Since $\eta_1 = \max(\rho(A_m^t))$ and $\eta_2 = \max(\rho(A_s^t))$, we have $\eta_1 < \eta_2$. From iteration (14), we have

$$\|\tilde{\Theta}^{T_0}\| \leq \eta_1^{T_0}\|\tilde{\Theta}^0\| + \frac{1-\eta_1^{T_0}}{1+\eta_1}b_0,$$

where $b_0 = \max_{i,j}\|\theta_i^* - \theta_j^*\|$. From (15), we have

$$\|\tilde{\Theta}_s^{T_0}\| \leq \eta_2^{T_0}\|\tilde{\Theta}^0\|.$$

From the Lipschitz continuous assumption on ∇f_i, we have $\|\tilde{\Theta}\| \geq \|\nabla f(\Theta)\|/\bar{L}_{f_i}$. Thus

$$\eta_1^{T_0} + \frac{(1-\eta_1^{T_0})b_0}{(1+\eta_1)\|\tilde{\Theta}^0\|} \leq \eta_1^{T_0} + \frac{(1-\eta_1^{T_0})b_0\bar{L}_{f_i}}{(1+\eta_1)\|\nabla f(\Theta^0)\|}.$$

By setting T_0 to satisfy

$$\eta_1^{T_0} + \frac{(1-\eta_1^{T_0})b_0\bar{L}_{f_i}}{(1+\eta_1)\|\nabla f(\Theta^0)\|} \leq \eta_2^{T_0},$$

we have

$$\eta_1^{T_0}\|\tilde{\Theta}^0\| + \frac{1-\eta_1^{T_0}}{1+\eta_1}b_0 < \eta_2^{T_0}\|\tilde{\Theta}^0\|,$$

which indicates $\|\tilde{\Theta}^{T_0}\| < \|\tilde{\Theta}_s^{T_0}\|$. After T_0, MTGD and single task gradient descent have same iterations. Since $\|\tilde{\Theta}^{T_0}\| < \|\tilde{\Theta}_s^{T_0}\|$, under same iterations, MTGD converges faster to the optimal solution than the single task gradient descent.

□

Remark 1. *To set T_0 satisfy condition* (12)*, the values of η_1 and η_2 are required. However, the exact expression of H^t is usually hard to be known. Denote $\xi = diag\{\xi_i\}$. Since $\lambda_i(\mathcal{M}^t - \alpha H^t) \leq \lambda_i(\mathcal{M}^t - \alpha\xi \otimes I_d)$ and $\lambda_i(I_{dT} - \alpha H^t) \leq \lambda_i(I_{dT} - \alpha\xi \otimes I_d)$, we can estimate η_1 and η_2 as $\eta_1 = \max\{\lambda_i(M^t - \alpha\xi)\}$ and $\eta_2 = \max\{\lambda_i(I_T - \alpha\xi)\}$.*

IV. Gradient-Free Multitasking Evolutionary Algorithm

A. OpenAI ES as Approximated Gradient Descent Iteration

The key point in Algorithm 1 is the update of the mean vector θ,

$$\theta^{t+1} = \theta^t - \alpha\frac{1}{n\sigma}\sum_{k=1}^{n} F_k\epsilon_k,$$

where $\epsilon_k \sim \mathcal{N}(0, I_d)$ and $F_k = F(\theta^t + \sigma\epsilon_k)$. The term $\frac{1}{n\sigma}\sum_{k=1}^{n} F_k\epsilon_k$ is in fact the estimated gradient of the expectation of F over ϵ. To show this under the isotropic multivariate Gaussian distribution with fixed covariance, note that

$$\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I_d)}F(\theta+\sigma\epsilon) = \mathbb{E}_{z\sim\mathcal{N}(\theta,\sigma^2 I_d)}F(z).$$

According to [26], using the "log-likelihood trick", we have

$$\begin{aligned} &\nabla_\theta\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I_d)}F(\theta+\sigma\epsilon) \\ =&\nabla_\theta\mathbb{E}_{z\sim\mathcal{N}(\theta,\sigma^2 I_d)}F(z) \\ =&\nabla_\theta\int F(z)g(z|\theta,\sigma)dz \\ =&\int F(z)\nabla_\theta g(z|\theta,\sigma)dz \\ =&\int F(z)\nabla_\theta\log g(z|\theta,\sigma)g(z|\theta,\sigma)dz \\ =&\mathbb{E}_{z\sim\mathcal{N}(\theta,\sigma^2 I_d)}[F(z)\nabla_\theta\log g(z|\theta,\sigma)] \\ =&\mathbb{E}_{z\sim\mathcal{N}(\theta,\sigma^2 I_d)}[F(z)\frac{z-\theta}{\sigma^2}] \\ =&\frac{1}{\sigma}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I_d)}[F(\theta+\sigma\epsilon)\epsilon], \end{aligned} \tag{17}$$

where $g(z|\theta,\sigma) = \frac{1}{\sqrt{(2\pi)^d}\sigma}e^{-\frac{(z-\theta)^T(z-\theta)}{2\sigma^2}}$. Since we sample n individuals in each generation, the expectation can be approximated as

$$\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I_d)}[F(\theta^t+\sigma\epsilon)\epsilon] \approx \frac{1}{n}\sum_{k=1}^{n} F_k\epsilon_k. \tag{18}$$

From Eq. (17) and Eq. (18), we have

$$\lim_{n\to\infty}\frac{1}{n\sigma}\sum_{k=1}^{n} F_k\epsilon_k = \nabla_\theta\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I_d)}F(\theta^t+\sigma\epsilon). \tag{19}$$

Precisely, the term $\frac{1}{n\sigma}\sum_{k=1}^{n} F_k\epsilon_k$ is asymptotically equivalent to the gradient of the expectation of F at θ^t as $n \to \infty$. Although F is usually assumed non-differentiable in evolutionary searches, $\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I_d)}F(\theta+\sigma\epsilon)$ is twice differentiable with respect to θ.

B. Multi-Task Evolution Strategies

Based on the MTGD and the asymptotic equivalence of OpenAI ES to gradient descent, a multi-task evolution strategies (MTES) inheriting the derived faster convergence results is proposed in Algorithm 2. In MTES, as shown in step 9, the update of the mean vector θ_i not only depends on its own information but also depends on information from related tasks, quantified by the transfer coefficient m_{ij}.

Algorithm 2 Pseudocode of MTES for Task i

Input: Objective function F_i, standard deviation σ, initial parameter θ_i^0 and $m_{ij}^0 = 0$ for $j \neq i$, learning rate α.

1: Set $t = 0$
2: **while** stopping criterion is not met **do**
3: **for** $k = 1, ..., n$ **do**
4: Draw sample $\epsilon_k \sim \mathcal{N}(0, I)$
5: Calculate fitness $F_{ik} = F_i(\theta_i^t + \sigma\epsilon_k)$
6: **end for**
7: Normalize fitness $N_{ik} = \frac{F_{ik} - \text{mean}(F_i)}{\text{std}(F_i)}$
8: Calculate transfer coefficient m_{ij}^t, $j \in \mathcal{N}_i$
9: $\theta_i^{t+1} = \sum_{j=1}^{T} m_{ij}^t \theta_j^t - \alpha\frac{1}{n\sigma}\sum_{k=1}^{n} N_{ik}\epsilon_k$
10: $t = t + 1$
11: **end while**

The paradigm of MTES considering two tasks i and j is shown in Fig. 1. As can be seen, two tasks are optimized simultaneously through two evolution strategies with independent populations. The distribution parameters of the populations are interacted through the transfer parameters m_{ij} and m_{ji} in the updates of the mean vectors of the populations, i.e., θ_i and θ_j. As depicted in Fig. 1, the stopping criterion can be determined by both tasks, i.e., stop after all the tasks are solved. However, if we care more about one task than others, the stopping criterion can be solely determined by the more focused task.

C. Calculating the Transfer Coefficient

The transfer coefficient m_{ij} determines the transfer strength of the information from task j to task i. If $m_{ij}^t = 0, j \neq i$, then there is no information transferred from task j to task i at iteration t. If $m_{ij}^t = m_{ji}^t \equiv 0$ and the stopping criterion is determined solely by each task, MTES separates into two base ESs. Since the update of θ is analogous to gradient descent, we give one simple but efficient scheme which uses the trajectory of θ to quantify the transfer coefficients. The pseudo code of calculating m_{ij}^t is outlined in Algorithm 3. The idea of Algorithm 3 is that if the mean of two populations are approaching, better solutions are believed to be located between the two populations. As a result, the mean vector of the other task will be utilized in the updating of the mean vector in the original task. In Algorithm 3, larger distance change results in larger transfer coefficients. Step 8 and Step 9 aim to normalize m_{ij} to satisfy condition (4). D^0 is the mean value of the initial distance between the mean vectors. It is used to adjust the relative importance of related tasks and the original task. It is worth noting that other sophisticated designs of the transfer coefficient can also be applied. As long as condition (4) is satisfied, the convergence property will not be violated.

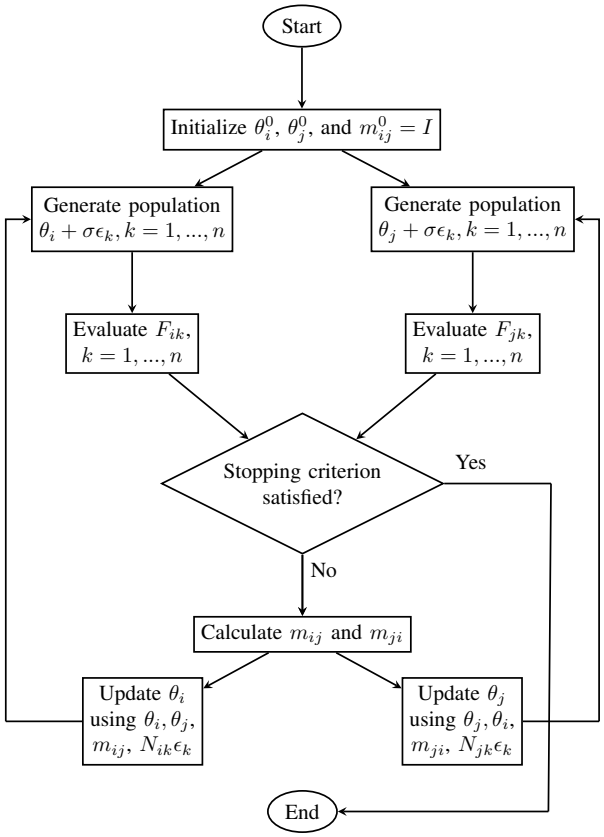


Fig. 1: MTES paradigm of two tasks.

Algorithm 3 Calculating Transfer Coefficient m_{ij}^t

Input: $\theta_i^{t-1}, \theta_i^t, \theta_j^{t-1}, \theta_j^t$
Output: m_{ij}^t

1: Calculate the Euclidean Distance between θ_i and θ_j at iteration t, i.e., $D_{ij}^t = \|\theta_i^t - \theta_j^t\|$. Set $D^0 = \sum_{j\in\mathcal{N}_i} D_{ij}^0/|\mathcal{N}_i|$.
2: Calculate the distance change $\Delta_{ij}^t = D_{ij}^{t-1} - D_{ij}^t$
3: **if** $\Delta_{ij}^t \leq 0$ **then**
4: $m_{ij}^t = 0$
5: **else**
6: $m_{ij}^t = \Delta_{ij}^t$
7: **end if**
8: Normalize $m_{ij}^t \to \frac{m_{ij}^t}{\sum_{j\in\mathcal{N}_i} m_{ij}^t + D^0}, j \in \mathcal{N}_i$
9: Calculate $m_{ii}^t = 1 - \sum_{j\in\mathcal{N}_i} m_{ij}^t$

TABLE II: Properties of single-objective continuous benchmark problem pairs.

Index	Category	Task	D	Search Space
1	CI+HS	Griewank (T1)	50	$[-100, 100]$
		Rastrigin (T2)	50	$[-50, 50]$
2	CI+MS	Ackley (T1)	50	$[-50, 50]$
		Rastrigin (T2)	50	$[-50, 50]$
3	CI+LS	Ackley (T1)	50	$[-50, 50]$
		Schwefel (T2)	50	$[-500, 500]$
4	PI+HS	Rastrigin (T1)	50	$[-500, 500]$
		Sphere (T2)	50	$[-50, 50]$
5	PI+MS	Ackley (T1)	50	$[-50, 50]$
		Rosenbrock (T2)	50	$[-50, 50]$
6	PI+LS	Ackley (T1)	50	$[-50, 50]$
		Weierstrass (T2)	25	$[-0.5, 0.5]$
7	NI+HS	Rosenbrock (T1)	50	$[-50, 50]$
		Rastrigin (T2)	50	$[-50, 50]$
8	NI+MS	Griewank (T1)	50	$[-100, 100]$
		Weierstrass (T2)	50	$[-0.5, 0.5]$
9	NI+LS	Rastrigin (T1)	50	$[-50, 50]$
		Schwefel (T2)	50	$[-500, 500]$

V. Experimental Results

In this section, experimental results on synthetic benchmark problems [27], the practical double-pole balancing controller design task with increasing challenging variants, and the parameterized planar arm problems are presented to show the efficacy of the proposed MTES. Since the evolution strategies provides candidate solutions, we treat θ as a candidate solution and draw $n-1$ ϵ_k in each generation to form the population of size n. The stopping criterion is set as all the tasks are either solved or terminated.

A. Synthetic Single-Objective Benchmark Functions

According to [27], nine single-objective continuous benchmark problems are used to test the proposed MTES against ES (Algorithm 1). TABLE II gives a summary of the properties of the nine problem pairs. The problem pairs are built by pairing classical single-objective functions considering the degree of intersection of the optima and inter-task similarity determined by Spearman's rank correlation. The degree of intersection can be classified into complete intersection (CI), partial intersection (PI), and no intersection (NI). The inter-task similarity can be classified into high similarity (HS), medium similarity (MS), and low similarity (LS). The dimensions of the tasks represented by D and the search space of each task are listed in TABLE II.

The settings in MTES and ES are the same. Specifically, the population size n is configured as 25 for each task and the maximum function evaluations is 250000 per task. The learning rate α and standard deviation σ are adjusted based on the value of θ. The initial values of α and σ are set as 0.1 for both tasks. For every 1000 generations, suppose the variation of θ is $\Delta_\theta = \theta^t - \theta^{t-1}$ and the median value of $|\Delta_\theta|$ is s, then σ is updated to be one times the order of magnitude of s if $s < 1$ and 1 otherwise. The step size α is updated as σ^2. The adapted σ and α make the perturbation scaled with the variation of θ and serve the same purpose as local search.

TABLE III gives the averaged objective function value obtained by the proposed MTES and ES over 30 independent runs. As can be seen from the results, MTES improves the results in most cases, which showcase the efficacy of MTES. The convergence behavior of MTES against ES is shown in Fig. 2 for five cases: 2, 5, 6, and 7. As is clear from Fig. 2, MTES converges faster than ES.

TABLE III: Objective function value of the continuous single-objective benchmark problem sets under MTES and ES. The results are averaged over 30 runs. The smaller the function values the better.

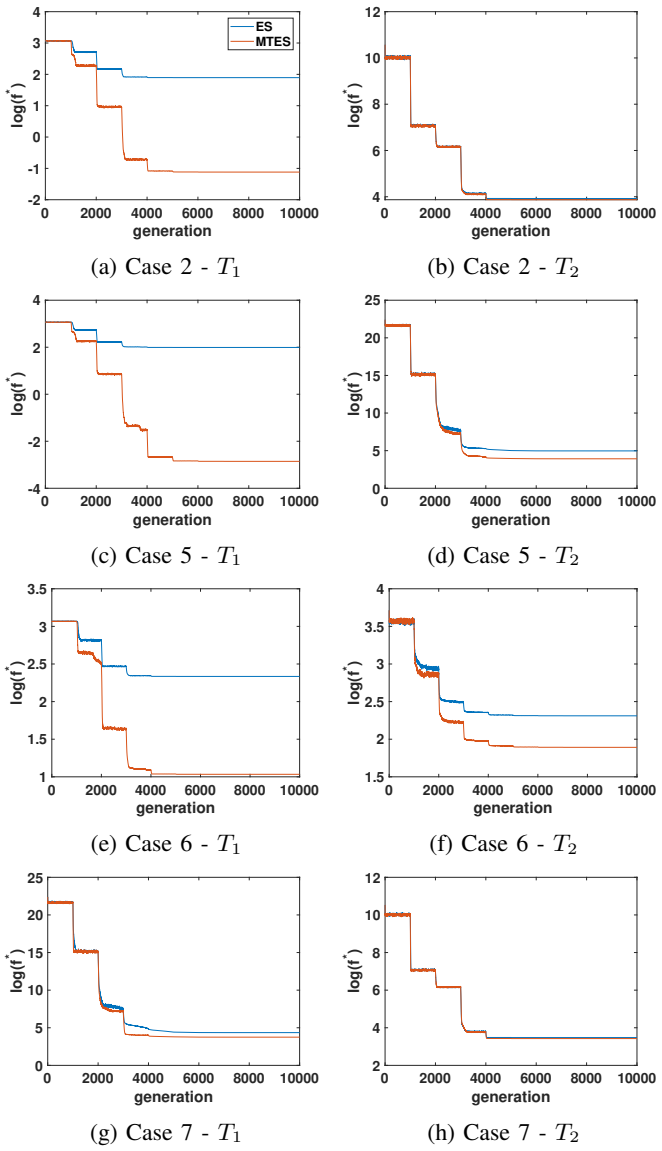
Index	ES		MTES	
	T1	T2	T1	T2
1	**1.20E-3**	3.37E+1	**1.20E-3**	**3.08E+1**
2	6.65E+0	5.06E+1	**3.27E-1**	**4.80E+1**
3	**2.00E+1**	**8.54E+3**	**2.00E+1**	9.49E+3
4	3.18E+1	**0.00E+0**	**3.06E+1**	**0.00E+0**
5	7.32E+0	1.44E+2	**5.76E-2**	**5.04E+1**
6	1.03E+1	1.01E+1	**2.81E+0**	**6.63E+0**
7	7.78E+1	3.24E+1	**4.25E+1**	**3.07E+1**
8	**0.00E+0**	1.80E+1	9.76E-2	**5.78E+0**
9	**3.06E+1**	**8.90E+3**	4.85E+3	1.33E+4

B. Double-Pole Balancing Problem

The pole balancing problem is a benchmark for artificial learning where neuroevolution-based approaches are used [10], [28], [29], [30]. The neuroevolution problems usually tend to be high-dimensional, multi-modal, but with highly redundant global optima, which are well-suited for ES to solve. The basic pole balancing system consists of a pole hinged to a cart moving on a finite track. To make the problem more challenging for neuroevolution-based approach, a variety of extensions to the basic pole balancing task have been made to produce variants with different difficulties, including add a second pole next to the other and restrict the amount of state information available to the controller. In this experiment, we use MTES to solve the double-pole balancing controller design task, with the complete state information (Markovian case) and incomplete state information where velocity information is absent (non-Markovian case).

The double-pole balancing problem involves balancing two different sized poles hinged on a cart that moves on a finite track by applying a continuous force to the cart, such that the poles do not fall beyond a predefined vertical angle. The relative lengths of the two poles determines the difficulty of the task. Generally, the length of the longer pole is fixed while the length of the shorter pole l_s is varied. The system becomes harder to control as l_s approaches to the length of the longer pole [28]. Considering multiple double-pole balancing tasks with different short pole lengths, since the tasks only differ in the short pole lengths, solving one task can provide useful information for other tasks, either from easier task to harder task, or vice versa. Thus, by combining the tasks with different short pole lengths together, MTES is used to solve multiple tasks simultaneously.

The equations of motion of the system and parameters are based on [28]. Forth-order Runge-Kuta integration with a step size of 0.01 second is used to implement the dynamics. A



(a) Case 2 - T_1 (b) Case 2 - T_2

(c) Case 5 - T_1 (d) Case 5 - T_2

(e) Case 6 - T_1 (f) Case 6 - T_2

(g) Case 7 - T_1 (h) Case 7 - T_2

Fig. 2: Convergence of the benchmark problems under MTES and ES. The value of the best performance in each generation is averaged over 30 runs.

neural network controller is used to output the force applied to the cart with the input being the state variables of the system. The optimization problem is to tune the weights of the neural network controller such that the outputted force solves the double-pole balancing task. The neural network controller receives the state variables scaled to $[-1.0, 1.0]$ and outputs a force between $[-10, 10]$N every 0.02 seconds. Fitness is determined by the number of time steps a network could keep both poles within a predefined angle and the cart between the ends of the track. A task is considered to be solved if the fitness is over 10^5 time steps. The track is set to 4.8m long and the predefined vertical angle within which the two poles are kept is set to $[-36°, 36°]$. The length of the longer pole is fixed to 1.00m. The initial position of the longer pole is set to be $1°$ from vertical direction and the shorter pole is set to be vertical.

1) Markovian Double-Pole Balancing: In the Markovian case, the state variables including the cart position, the cart velocity, the angles of the two poles, and the angular velocities of the two poles are available as the input of the neural network controller. A simple feedforward neural network with ten hidden neurons is used as the controller. As a result, there are 70 weights need to be optimized. The population size is set to 20 and the maximum evaluation times is set to 10000 for each task. The step size α and the standard deviation σ are both set to 0.5 via an empirical investigation.

To test MTES in solving multiple tasks simultaneously, three tasks of different short pole lengths are considered and different task pairs as in [10] are used. The percentage of success runs over 30 independent runs is used as a performance measure. To showcase the facet of multitasking, the results obtained by ES which solves one task in a single run are listed as a comparison. In addition, the results published in [10] are listed in TABLE IV for comparison.

From the results, we can see that MTES performs best among the three algorithms. Specifically, comparing the results obtained by MTES in task pair (T_2, T_3) and ES for tasks T_2 and T_3 separately, it can be seen that by tackling the tasks in a multitasking environment, not only the success rate for the harder task T_3 improves from 70.0% to 93.3%, the success rate of the easier task T_2 also improves from 96.7% to 100%. This showcases the distinction between multitasking and sequential transfer learning where only the targeted task is beneficial. The result for the three tasks together (T_1, T_2, T_3) also shows that both the harder and the easier tasks are beneficial from the multitasking environment, where useful information from similar tasks helps boosting the performance.

We extend the experiment to solving six different tasks simultaneously as in [10], where the short pole lengths ranging from 0.45m to 0.70m. The percentage of success times out of 30 runs is summarized in TABLE V. The result of MFEA-II is obtained from [10]. Overall, the results reveal that MTES outperforms other methods in comparison. With the transfer incorporated into the single ES, the improved results of MTES over ES confirmed the effectiveness of conducting multitasking. As the length of the short pole l_s increases, the task becomes harder to solve. While MFEA-II and MTES have shown similar performance on the first three tasks, MTES gradually outperforms MFEA-II on harder tasks T_4 to T_6.

2) Non-Markovian Double-Pole Balancing: In this section, we solve the double-pole balancing problem without velocity information, i.e., only the position of the cart and the angle of the two poles are feed into the neural network.

This problem is much harder to solve since without the velocity information, the network needs to be recurrent so that the velocities can be computed internally using feedback connections. To prevent the system from solving the task simply by moving the cart back and forth quickly without calculating the velocities, the fitness which penalizes oscillations introduced by [31] is used, which is the weighted sum of two fitness component functions taken over 1000 time steps:

TABLE IV: Success rate of three different tasks and task pairs of the Markovian double-pole balancing problem. The best results are highlighted in bold. The dash symbol implies the absence of a task in the multitasking algorithms.

Task	l_s	ES	(T_1, T_2)		(T_1, T_3)		(T_2, T_3)		(T_1, T_2, T_3)	
			MTES	MFEA-II	MTES	MFEA-II	MTES	MFEA-II	MTES	MFEA-II
T_1	0.60m	**100%**	**100%**	30%	**100%**	30%	-	-	**100%**	47%
T_2	0.65m	96.7%	**100%**	27%	-	-	**100%**	27%	**100%**	37%
T_3	0.70m	70.0%	-	-	**86.7%**	7%	**93.3%**	27%	**90.0%**	17%

TABLE V: Success rate of six different tasks of the Markovian double-pole balancing problem. The six tasks are solved simultaneously by multitasking algorithms MTES and MFEA-II.

Task	l_s	ES	MTES	MFEA-II
T_1	0.45m	**100%**	**100%**	**100%**
T_2	0.50m	**100%**	**100%**	**100%**
T_3	0.55m	**100%**	**100%**	97%
T_4	0.60m	**100%**	**100%**	83%
T_5	0.65m	96.7%	**100%**	63%
T_6	0.70m	70.0%	**96.7%**	37%

$$F = 0.1f_1 + 0.9f_2,$$
$$f_1 = t/1000,$$
$$f_2 = \begin{cases} 0 & \text{if } x < 0 \\ \frac{0.75}{\sum_{i=t-100}^{t}(|x^i|+|\dot{x}^i|+|\theta_{pl}^i|+|\dot{\theta}_{pl}^i|)} & \text{otherwise,} \end{cases}$$

where t is the number of time steps the poles remain balanced during the 1000 time steps, x is the cart position, $\dot{x}$ is the velocity of the cart, θ_{pl} is the angle of the longer pole, and $\dot{\theta}_{pl}$ is the angular velocity of the longer pole. Following [31], the generalization test is conducted for the sake of robustness. In addition to balancing the poles for 1000 time steps, a controller is tested to balance the poles from 625 different initial states, each for 1000 time steps. If the network can generalize to at least 200 of the 625 initial states, the network is counted as a solution. The start values of the state variables including the cart position, the cart velocity, the longer pole angle, and the longer pole angular velocity are varied systematically within the following values (-1.944, -1.08, 0.0, 1.08, 1.944)m, (-1.215, -0.675, 0.0, 0.675, 1.215)m/s, (-0.056548, -0.031416, 0.0, 0.031416, 0.056548)rad, and (-0.135088, -0.075049, 0.0, 0.075049, 0.135088)rad/s, respectively. The initial states of the short pole angle and angular velocity are set to zero.

A recurrent neural network with three hidden neurons is used as the controller. The population size and the maximum evaluation times are set the same as the Markovian case, which are 20 and 10000 for each task, respectively. The step size α is set to 0.1 and the standard deviation σ is set to 0.2.

Ten tasks that the short pole lengths vary from 0.10m to 0.55m are considered. The consecutive two tasks are paired together to be solved by MTES and MFEA-II, and each task is solved independently by ES for comparison. For the state-of-the-art multitasking algorithm MFEA-II, the same recurrent neural network with three hidden neurons is used. Using the default setting from the authors, which has the same evaluation times with MTES, the results obtained by MFEA-II show that MFEA-II is not effective in solving the non-Markovian double-pole balancing problem, which shows the difficulty of this problem. TABLE VI shows the success rates over 10 independent runs of ES and MTES.

TABLE VI: Success rate of non-Markovian double-pole balancing tasks with increasing short pole lengths. Consecutive two tasks are paired together to test the MTES.

Task Pairs	Task	l_s(m)	ES	MTES
(T_1, T_2)	T_1	0.10	**100%**	**100%**
	T_2	0.15	**100%**	**100%**
(T_3, T_4)	T_3	0.20	90%	**100%**
	T_4	0.25	**100%**	**100%**
(T_5, T_6)	T_5	0.30	**100%**	**100%**
	T_6	0.35	90%	**100%**
(T_7, T_8)	T_7	0.40	90%	**100%**
	T_8	0.45	80%	**100%**
(T_9, T_{10})	T_9	0.50	60%	**100%**
	T_{10}	0.55	20%	**90%**

According to TABLE VI, it can be seen that by solving different tasks simultaneously using MTES, the performance is equal to or better than tackling the tasks separately using ES. Specifically, the success rates for both tasks are improved for task pairs (T_7, T_8) and (T_9, T_{10}) compared to solving the four tasks separately. For more challenging tasks T_9 and T_{10}, significant improvements can be observed where the success rates improved from 60% and 20% to 100% and 90%, respectively. Then, we added two more challenging tasks T_{11} and T_{12} with short pole lengths being 0.60m and 0.65m, respectively. The success rates of ES solving T_{11} and T_{12} individually and MTES solving (T_{11}, T_{12}) as a pair are all zero. We test the case that 6 harder tasks $T_7 - T_{12}$ are solved simultaneously using MTES. The results are listed in TABLE VII. As can be seen from the results, when solving the six tasks together, the success rate for task T_{11} in MTES improves from 0% to 20%, while the success rate for task T_{10} decreases from 90% to 80%, compared to solving (T_9, T_{10}) together. This phenomenon verifies that knowledge in simpler tasks can help to find solutions for the harder task, while the mutual transfer may bring negative transfer for the simpler task.

To show the efficiency of the proposed MTES, the evaluations used to find the solution and the generalization performance are compared to existing systems which have been demonstrated able to solve the non-Markovian double-pole balancing task: NeuroEvolution of Augmenting Topologies (NEAT) [29], Cellular Encoding (CE) [32], and Enforced Subpopulation (ESP) [33]. The compared results in TABLE VIII are published in [29] for the task with the short

TABLE VII: Success rate of six harder tasks $T_7 - T_{12}$ of the non-Markovian double-pole balancing problem. The six tasks are solved simultaneously by multitasking algorithm MTES. N.A. implies no solution obtained within the maximum evaluation times.

Task	l_s(m)	ES	MTES
T_7	0.40	90%	**100%**
T_8	0.45	80%	**100%**
T_9	0.50	60%	**100%**
T_{10}	0.55	20%	**80%**
T_{11}	0.60	N.A.	**20%**
T_{12}	0.65	N.A.	N.A.

TABLE VIII: Efficiency for non-Markovian double-pole balancing task with $l_s = 0.1$m. The results are averaged over 20 runs. The generalization results are out of 625 cases in each simulation.

Method	Evaluations	Generalization
CE	840,000	**300.0**
ESP	169,466	289.0
NEAT	33,184	286.0
MTES	**1,754**	278.2

pole length being 0.1m. In MTES, we solve the task pair (T_1, T_2) and show the respective results for T_1. TABLE VIII shows that MTES is the fastest method on this challenging task. It requires minimum number of evaluations, while the generalization performance of MTES is comparable to the comparing algorithms. The evaluations and generalization are also compared between MTES and ES for the ten tasks $T_1 - T_{10}$. Since in the multi-task environment, the variables of the tasks are updated until all the tasks are being solved or terminated, the evaluation times of the two tasks in the same pair in MTES are the same. The results in TABLE IX show that MTES requires fewer evaluations in most tasks except T_1 and T_9. Actually, the evaluations used for finding a solution for T_9 are $6,940$, the extra evaluations are used to facilitate the solving of T_{10}. Nevertheless, compare the total evaluations used by T_1 and T_2 in MTES and ES, MTES still requires fewer evaluations, which is also the case for tasks T_9 and T_{10}. Furthermore, MTES has better generalization performance in most tasks except T_1 and T_2, which show the strength of MTES in solving harder tasks.

C. Parameterized Planar Arm

In this section, we test the MTES on the parameterized planar kinematic arm problem adopted from [34]. A planar kinematic arm is an arm consist of d joints and $d+1$ links that moves in a plane. The objective of the task is to find the angle of each joint such that the tip of the arm is as close as possible to a predefined target in the plane. By parameterizing the arm with the length of the links, L, and the maximum angle for each joint, $\alpha_{\max}$ (all the links and joints have the same length and angle limits), we can produce different tasks. To be precise, a task is defined by a particular combination of L and $\alpha_{\max}$, a candidate solution $\theta \in \mathbb{R}^d$ is a d-dimensional vector composed of the angle of each joint. The fitness function f is the Euclidean distance from the tip position to the target position. More details of the kinematics of the arm can refer to [34].

TABLE IX: Efficiency for non-Markovian double-pole balancing task with increasing l_s. The results are averaged over 10 runs.

Task	Evaluations		Generalization	
	ES	MTES	ES	MTES
T_1	**1,660**	1,754	**279.0**	278.2
T_2	2,270	**1,754**	**304.9**	280.2
T_3	3,804	**1,982**	247.7	**299.4**
T_4	2,528	**1,982**	286.2	**296.6**
T_5	3,144	**2,944**	267.4	**279.4**
T_6	4,650	**2,944**	222.9	**249.8**
T_7	5,358	**4,828**	222.2	**246.2**
T_8	5,702	**4,828**	197.0	**234.6**
T_9	**7,256**	8,460	144.2	**225.6**
T_{10}	9,694	**8,460**	41.0	**200.1**

TABLE X: Success rate of planar arm tasks with different link lengths and angle limits. Two sets of six tasks are joined together to test the MTES.

Task Groups	Task	L(m)	$\alpha_{\max}$(rad)	ES	MTES
$T_1 - T_6$	T_1	0.10	0.80	**100.00%**	**100.00%**
	T_2	0.10	0.75	90.00%	**100.00%**
	T_3	0.10	0.70	70.00%	**100.00%**
	T_4	0.10	0.65	76.67%	**100.00%**
	T_5	0.10	0.60	53.33%	**100.00%**
	T_6	0.10	0.55	23.33%	**73.33%**
$T_7 - T_{12}$	T_7	0.08	0.40	96.67%	**100.00%**
	T_8	0.08	0.35	56.67%	**100.00%**
	T_9	0.07	0.40	96.67%	**100.00%**
	T_{10}	0.07	0.35	93.33%	**100.00%**
	T_{11}	0.06	0.40	96.67%	**100.00%**
	T_{12}	0.06	0.35	50.00%	**93.33%**

In this experiment, we set the dimensionality d as 10 and the target position as (0.5,0.5). By varying the link length L and the angle limits $\alpha_{\max}$, we define 12 tasks, where $T_1 - T_6$ have the same total link length of 1m and varying angle limits ranging from 0.80rad to 0.55rad. $T_7 - T_{12}$ have varying link lengths for different task pairs, and the angle limits for different task pairs are kept the same. The details of the tasks can be seen in TABLE X. We set the total evaluation times as 10000 for each task. To use MTES, first, we solve tasks $T_1 - T_6$ together, and then solve $T_7 - T_{12}$ together. As a comparison, $T_1 - T_{12}$ are solved individually using ES. A task is regarded as success if the value of fitness function is below 0.2. We show the success rate obtained by MTES and ES over 30 runs in TABLE X.

As can be seen from TABLE X, solving a group of tasks simultaneously using MTES achieves better performance than solving the tasks individually using ES. For tasks T_1 to T_6, where the link lengths L are the same, the smaller the angle limit $\alpha_{\max}$, the lower the success rate achieved by ES, while the success rate achieved by MTES remains relatively high compared to ES, which shows the capability of MTES in solving harder tasks. The tasks $T_7 - T_{12}$ in the second group have varying lengths and fixed angle limits for different task pairs, and different angle limits for the tasks within each task

pair. As can be seen, both the link length and the angle limit affect the success rate. The influence of the link length may depend on the target position, the smaller the angle limit, the harder the task, and the angle limit has a larger influence than the link length. When the angle limits change from 0.4 to 0.35, the performance of ES drops a lot, while MTES maintains high success rates. The convergence of ES and MTES for the 12 tasks is plotted in Fig. 3. It can be seen that MTES converges faster in most cases and achieves a better result in all the tasks within a limited generation.

VI. Conclusion

In this paper, we proposed a novel multi-task gradient descent (MTGD) algorithm and extended it to a novel gradient-free evolutionary multitasking algorithm, multi-task evolution strategies (MTES), capable of solving multiple tasks simultaneously while benefiting from related tasks. In contrast to existing evolutionary multitasking algorithms, the influence of incorporating information from other tasks has been theoretically analyzed based on the gradient descent iteration analysis. We found out that the convergence of single task gradient descent will not be impeded with transferring information from other tasks as long as the transfer coefficients satisfying certain conditions. This allows a flexible design of transferring mechanisms. Furthermore, for symmetric transferring between two tasks, we have proven that the convergence rate of MTGD is faster than single task gradient descent under certain conditions. To experimentally validate this result and to test its implications for the case of gradient-free evolutionary multitasking, MTES has been used to solve the synthetic multi-task benchmark functions, the practical double-pole balancing problems (both the Markovian and non-Markovian cases), and the parameterized planar arm problems under a multi-task environment. The results obtained have been compared to the corresponding single task ES, which confirmed the efficacy of the proposed MTES. The comparison between MTES and other algorithms that are used to solve the double-pole balancing problem also show the efficiency of the proposed algorithm in solving practical optimization tasks.

References

[1] M. Cavazzuti, *Optimization methods: from theory to design scientific and technological aspects in mechanics*. Springer Science & Business Media, 2012.

[2] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, "A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 321–344, 2012.

[3] A. S. Azad, M. Islam, and S. Chakraborty, "A heuristic initialized stochastic memetic algorithm for MDPVRP with interdependent depot operations," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4302–4315, 2017.

[4] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for machine learning*. Mit Press, 2012.

[5] A. Gupta, Y.-S. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 51–64, 2017.

[6] X. Zhang, Y. Zhuang, W. Wang, and W. Pedrycz, "Transfer boosting with synthetic instances for class imbalanced object recognition," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 357–370, 2016.

[7] L. Bai, Y.-S. Ong, T. He, and A. Gupta, "Multi-task gradient descent for multi-task learning," *Memetic Computing*, vol. 12, no. 4, pp. 355–369, 2020.

[8] Y.-S. Ong and A. Gupta, "Air 5: Five pillars of artificial intelligence research," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 5, pp. 411–415, 2019.

[9] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2015.

[10] K. K. Bali, Y.-S. Ong, A. Gupta, and P. S. Tan, "Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II," *IEEE Transactions on Evolutionary Computation*, 2019.

[11] A. Gupta, Y.-S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1652–1665, 2016.

[12] K. K. Bali, A. Gupta, Y.-S. Ong, and P. S. Tan, "Cognizant multitasking in multiobjective multifactorial evolution: MO-MFEA-II," *IEEE Transactions on Cybernetics*, 2020.

[13] Y.-W. Wen and C.-K. Ting, "Parting ways and reallocating resources in evolutionary multitasking," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2404–2411, IEEE, 2017.

[14] T. P. Dinh, B. H. T. Thanh, T. T. Ba, and L. N. Binh, "Multifactorial evolutionary algorithm for solving clustered tree problems: competition among cayley codes," *Memetic Computing*, vol. 12, no. 3, pp. 185–217, 2020.

[15] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y.-S. Ong, K.-C. Tan, and A. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE Transactions on Cybernetics*, vol. 49, no. 9, pp. 3457–3470, 2018.

[16] G. Li, Q. Lin, and W. Gao, "Multifactorial optimization via explicit multipopulation evolutionary framework," *Information Sciences*, vol. 512, pp. 1555–1570, 2020.

[17] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with tsp, qap, lop, and jsp," in *2016 IEEE Region 10 Conference (TENCON)*, pp. 3157–3164, IEEE, 2016.

[18] E. Osaba, A. D. Martinez, A. Galvez, A. Iglesias, and J. Del Ser, "dMFEA-II: An adaptive multifactorial evolutionary algorithm for permutation-based discrete optimization problems," *arXiv preprint arXiv:2004.06559*, 2020.

[19] L. Feng, L. Zhou, A. Gupta, J. Zhong, Z. Zhu, K.-C. Tan, and K. Qin, "Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking," *IEEE Transactions on Cybernetics*, 2019.

[20] 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*, pp. 1–26, 2020.

[21] R. Chandra, Y.-S. Ong, and C.-K. Goh, "Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction," *Neurocomputing*, vol. 243, pp. 21–34, 2017.

[22] J. Liang, K. Qiao, M. Yuan, K. Yu, B. Qu, S. Ge, Y. Li, and G. Chen, "Evolutionary multi-task optimization for parameters extraction of photovoltaic models," *Energy Conversion and Management*, vol. 207, p. 112509, 2020.

[23] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[24] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing*, vol. 3, pp. 323–453, Elsevier, 2014.

[25] R. A. Horn, R. A. Horn, and C. R. Johnson, *Matrix analysis*. Cambridge university press, 1990.

[26] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 949–980, 2014.

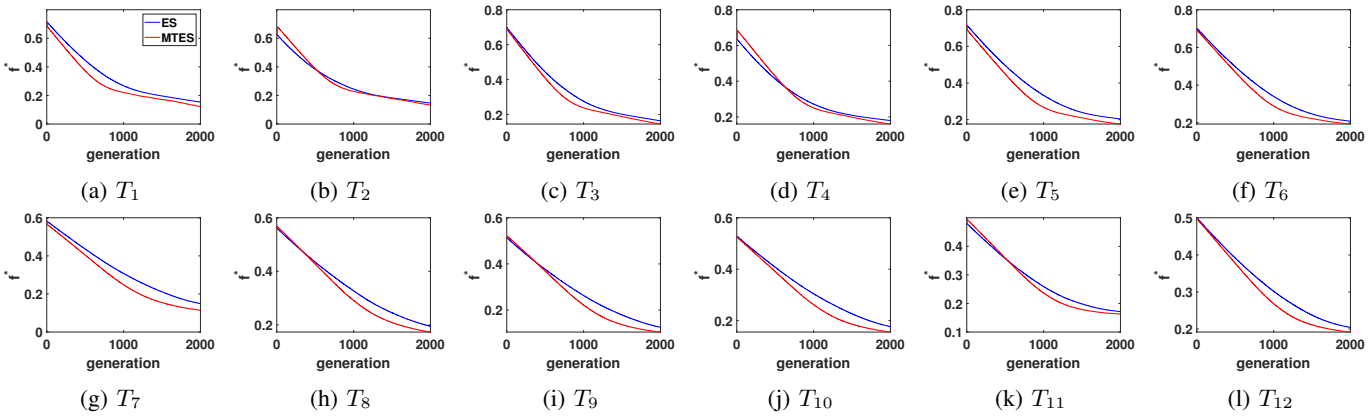
[27] B. Da, Y.-S. Ong, L. Feng, A. K. Qin, A. Gupta, Z. Zhu, C.-K. Ting, K. Tang, and X. Yao, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," *Technical Report*, 2016.

[28] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated neural evolution through cooperatively coevolved synapses," *Journal of Machine Learning Research*, vol. 9, no. May, pp. 937–965, 2008.

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

[30] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber, "Natural evolution strategies," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3381–3387, IEEE, 2008.

[31] F. Gruau, D. Whitley, and L. Pyeatt, "A comparison between cellular encoding and direct encoding for genetic neural networks," in *Proceedings of the 1st annual conference on genetic programming*, pp. 81–89, 1996.



(a) T_1 (b) T_2 (c) T_3 (d) T_4 (e) T_5 (f) T_6

(g) T_7 (h) T_8 (i) T_9 (j) T_{10} (k) T_{11} (l) T_{12}

Fig. 3: Convergence of the parameterized planar arm problem under MTES and ES. The value is averaged over 30 runs.

[32] F. Gruau, "Genetic synthesis of modular neural networks," in *Proceedings of the 5th International Conference on Genetic Algorithms*, (San Francisco, CA, USA), p. 318–325, Morgan Kaufmann Publishers Inc., 1993.

[33] F. J. Gomez and R. Miikkulainen, "Solving non-markovian control tasks with neuroevolution," in *IJCAI*, vol. 99, pp. 1356–1361, 1999.

[34] J.-B. Mouret and G. Maguire, "Quality diversity for multi-task optimization," *arXiv preprint arXiv:2003.04407*, 2020.

Lu Bai received the B.Eng. and M.Eng. degrees in automation from Xiamen University, Xiamen, China, in 2012 and 2015, respectively, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2020. She is currently a Research Fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. Her research interests include distributed control and optimization, multi-task optimization, and multi-task learning.

Wu Lin received the B.S. degree from Hubei University of Technology, Wuhan, China, in 2017, and received the M.S. degree from Shenzhen University, Shenzhen, China, in 2020. He is currently a Research Assistant in College of Computer Science and Software Engineering, Shenzhen University. His current research interests are in evolutionary computation, multimodal multi-objective optimization, machine learning and transfer optimization.

Abhishek Gupta received his PhD in Engineering Science from the University of Auckland, New Zealand, in the year 2014. He currently serves as a Scientist and Technical Lead in the Singapore Institute of Manufacturing Technology (SIMTech), at the Agency for Science, Technology and Research (A*STAR), Singapore. He has diverse research experience in the field of computational science, ranging from numerical methods in engineering physics, to topics in computational intelligence. His current research interests lie in probabilistic model-based search algorithms with transfer and multitask learning capabilities, for applications spanning cyber physical production systems and engineering design.

Yew-Soon Ong (M'99-SM'12-F'18) received the Ph.D. degree in artificial intelligence in complex design from the University of Southampton, U.K., in 2003. He is President's Chair Professor in Computer Science at Nanyang Technological University (NTU), and holds the position of Chief Artificial Intelligence Scientist of the Agency for Science, Technology and Research Singapore. At NTU, he serves as Director of the Data Science and Artificial Intelligence Research and co-Director of the Singtel-NTU Cognitive & Artificial Intelligence Joint Lab. His research interest is in artificial and computational intelligence. He is founding Editor-in-Chief of the IEEE Transactions on Emerging Topics in Computational Intelligence and AE of IEEE Transactions on Neural Networks & Learning Systems, IEEE Transactions on Cybernetics, IEEE Transactions on Artificial Intelligence and others. He has received several IEEE outstanding paper awards and was listed as a Thomson Reuters highly cited researcher and among the World's Most Influential Scientific Minds.