Meta-Learning Inspired Single-Step Generative Model for Expensive Multitask Optimization Problems

Ruilin Wang, Xiang Feng*, Huiqun Yu, *Senior Member, IEEE*, Yang Tan, Edmund M-K Lai, *Life Senior Member, IEEE*

***Abstract*—In expensive multitask optimization problems (ExMTOPs), multiple complex tasks must be optimized simultaneously under limited computational budgets. Existing approaches, often based on surrogate models, aim to approximate objective functions but struggle to generalize across heterogeneous tasks, depend on task-specific sampling, and require frequent retraining. To address these challenges, we propose the Multifactorial Evolutionary Algorithm–Single Step Generative Model (MFEA-SSG), a meta-learning-inspired framework that learns to generate high-quality solutions across tasks. Inspired by meta-learning, we treat each random shuffle of the decision variables as a unique pseudo-task, training the model on a distribution of these tasks to learn a task-agnostic prior about the structure of elite solutions. This process disrupts task-specific dependencies, allowing the model to learn transferable structures from recomposed samples. We then adopt a diffusion-based generative model to learn the distribution of optimal solutions, enabling knowledge transfer across tasks without directly approximating objective functions. To reduce inference cost, we introduce a student model distilled from the diffusion process. Unlike conventional diffusion models that denoise iteratively, the student generates solutions in a single forward pass, significantly reducing inference time. Comprehensive experiments on both general multitask benchmarks and a real-world protein mutation prediction scenario demonstrate that MFEA-SSG achieves high-quality solutions with fast convergence and low computational cost under limited evaluation budgets, outperforming state-of-the-art general and ExMTOPs algorithms.**

***Index Terms*—Expensive evolutionary multitasking, multifactorial evolutionary algorithm, meta learning, generative model, knowledge transfer.**

I. Introduction

EVOLUTIONARY multitasking (EMT) is an emerging optimization paradigm aimed at simultaneously solving multiple optimization tasks by sharing knowledge across tasks, thus enhancing optimization efficiency [1]–[3]. Traditional EMT methods perform well in handling tasks with inter-task similarity, leveraging shared knowledge to significantly accelerate convergence [4], [5]. However, in practical applications, many optimization tasks involve expensive evaluations, such as complex engineering designs or high-cost experiments, where each evaluation demands substantial computational resources [6]. These expensive multitask optimization problems (ExMTOPs) challenge the applicability of traditional EMT, as they typically require numerous function evaluations to ensure solution quality. Given the often limited computational resources, obtaining high-quality solutions within a constrained number of evaluations has become a substantial challenge for EMT. These challenges are further exacerbated in heterogeneous multitask scenarios, where the constituent tasks exhibit significant differences in their characteristics, such as having dissimilar fitness landscapes, mismatched optimal locations, or low inter-task correlation. Such heterogeneity can often lead to negative transfer in traditional EMT algorithms.

A number of recent efforts have explored surrogate-based frameworks to address the challenges of ExMTOPs [7]–[11]. A primary strategy involves using surrogate models to guide the search, for instance, employing radial basis functions to steer adaptive sampling based on inter-task similarity [12] or to reduce evaluations in complex settings like minimax optimization [13]. More advanced frameworks focus on sophisticated knowledge transfer mechanisms. For example, Tan et al. proposed a two-phase approach combining a multitask Gaussian Process model for global knowledge transfer with Bayesian optimization for local refinement [6]. Pushing this transfer learning concept further, ExTrEMO utilizes a factorized Transfer Gaussian Process, enabling it to leverage knowledge from a wide array of source tasks—including negatively correlated ones—to accelerate convergence, particularly in expensive multiobjective settings [14]. Distinct from these forward-modeling approaches, another emerging paradigm is inverse modeling, which maps from the objective space back to the decision space. This has been explored through Bayesian inverse transfer for handling tasks with heterogeneous decision spaces [15], and through conditional generative models that can be "prompted" with desired performance criteria to generate corresponding solutions [16]. This concept has been extended to classifier-guided diffusion models, where a classifier trained on evaluated solutions guides the generation of high-quality candidates for complex design problems like robot morphology [17]. In parallel to these modeling advancements, other research has focused on improving the implementation and adaptability of ExMTOPs. Huang et al. developed a

This work is supported by the National Natural Science Foundation of China (No.62276097), Key Program of National Natural Science Foundation of China (No.62136003), National Key Research and Development Program of China (No.2020YFB1711700), Special Fund for Information Development of Shanghai Economic and Information Commission (No.XX-XXFZ-02-20-2463), Scientific Research Program of Shanghai Science and Technology Commission (No.21002411000) and the China Scholarship Council (No. 202406740010). The authors would like to thank Shenglian Tan from Central South University for providing essential data support. *(Corresponding author: Xiang Feng.)*

R. Wang, X. Feng, and H. Yu are with the School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China, and also with the Shanghai Engineering Research Center of Smart Energy, Shanghai, China (e-mail: wrl61@mail.ecust.edu.cn; xfeng@ecust.edu.cn; yhq@ecust.edu.cn).

Y. Tan is with the School of Computer Science, Shanghai Jiao Tong University, Shanghai, China (e-mail: tanyang.august@sjtu.edu.cn).

E. M-K. Lai is with the Department of Data Science and Artificial Intelligence, Auckland University of Technology, Auckland, New Zealand (e-mail: edmund.lai@aut.ac.nz).

GPU-based framework to accelerate the optimization process by handling a large number of tasks simultaneously through parallel computing [18], while other work has also focused on developing adaptive knowledge transfer mechanisms to improve overall search efficiency [19].

While surrogate models offer advantages in ExMTOPs [20], they also have several limitations. Although their goal is to guide the search by capturing the global trends of the fitness landscape, their reliability can be compromised in high-dimensional spaces under a severely limited evaluation budget [21]. In such data-scarce scenarios, the model faces an increased risk of either failing to capture these crucial trends or overfitting to sparse local data, which can lead to misguided exploration and necessitate frequent retraining. Furthermore, surrogate models are typically crafted for single tasks or narrow problem classes, limiting their adaptability and reducing the effectiveness of knowledge transfer in multitask settings. The presence of complex interdependencies among decision variables further degrades their predictive performance as dimensionality increases. These limitations hinder the applicability of surrogate-based methods in high-dimensional, multitask, and evaluation-limited scenarios.

To address these issues, we propose a generative-model-based algorithm tailored for ExMTOPs. Our approach is inspired by the core principle of meta-learning: learning to generalize from a distribution of related tasks. We simulate a distribution of pseudo-tasks by randomly shuffling the dimensions of the decision variables. Each shuffle creates a new learning problem, compelling the model to learn a dimension-agnostic, transferable prior about the structure of high-quality solutions, rather than overfitting to an arbitrary, fixed variable ordering. Unlike surrogate models that approximate objective values, our approach focuses on learning the distribution of promising solutions directly. Specifically, the algorithm samples individuals from tasks, shuffles and reshapes them into image representations, and trains a diffusion-based generative model to capture underlying structural patterns. To reduce inference cost, we further apply knowledge distillation to train a lightweight student model that approximates the end-to-end generation of the complex, multi-step diffusion model. This student model, which serves as our efficient "single-step generative model", is capable of producing high-quality individuals in a single forward pass, i.e., without the need for iterative denoising steps. This single-step generative model enables fast inference with minimal computational overhead. Our contributions can be summarized as follows:

1) We propose a generative-model-based framework that significantly enhances sample efficiency in ExMTOPs. It achieves this by directly learning the distribution of high-quality solutions—a more direct and feasible task in data-scarce scenarios than approximating the entire fitness landscape. By generating new candidates from a progressively refined model of these elite solutions, our framework inherently biases the search toward promising regions, thereby minimizing wasted function evaluations and achieving superior performance under strict evaluation budgets.
2) We introduce a meta-learning-inspired mechanism that enhances generalization by randomly shuffling the dimensions of decision variables and reshaping them into image-like representations. This disrupts task-specific dependencies and avoids artificial spatial biases introduced by naive reshaping, allowing the model to learn transferable patterns from recomposed samples and generalize across diverse tasks.
3) We apply knowledge distillation to derive a lightweight student model that enables single-step sample generation—i.e., generating high-quality individuals in a single forward pass without iterative denoising. Both theoretical analysis and empirical results confirm that the student model significantly reduces inference cost while preserving optimization performance.

II. PRELIMINARY

A. Multifactorial Evolution Algorithm

EMT aims to solve multiple optimization tasks concurrently by enabling knowledge transfer across tasks. In ExMTOPs, the same goal is pursued under strict evaluation budget constraints, where only a limited number of function evaluations are allowed for each task. Formally, given T tasks $\{T_i\}_{i=1}^{T}$, where each task T_i is defined as minimizing an objective function f_i over a decision space X_i, the overall optimization objective can be formally defined as:

$$\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_T^*\} = \{\arg\min_{\mathbf{x}_1 \in X_1} f_1(\mathbf{x}_1), \arg\min_{\mathbf{x}_2 \in X_2} f_2(\mathbf{x}_2), \ldots, \arg\min_{\mathbf{x}_T \in X_T} f_T(\mathbf{x}_T)\} \tag{1}$$

To address the multitask optimization problem, the pioneering framework is the Multifactorial Evolutionary Algorithm (MFEA) [22]. As the first to introduce the concept of evolutionary multitasking, MFEA formally defined it as a mathematical optimization problem and provided the foundational approach for solving it. It treats each task as an independent environment and maintains a shared population where each individual is associated with a skill factor indicating the task it performs best on. MFEA quantifies individual performance through factorial cost (objective value plus penalties) and factorial rank (its position among task-specific solutions), and derives a scalar fitness (the reciprocal of the best rank) for selection across tasks. This enables implicit knowledge transfer, allowing advantageous traits discovered in one task to benefit others.

The MFEA workflow begins by generating a unified population, evaluating individuals across all tasks, assigning skill factors, and applying assortative mating—promoting task-specific exploitation while enabling occasional inter-task crossover. Offspring are generated through simulated binary crossover and polynomial mutation, then evaluated selectively on their assigned tasks to reduce evaluation cost. Scalar fitness is used to rank individuals, and the fittest are carried into the next generation. As a foundational framework for evolutionary multitasking, MFEA has shown remarkable capability in leveraging shared genetic structure across tasks. However, when

applied to ExMTOPs, where function evaluations are severely limited, its performance degrade due to inefficient sampling or excessive reliance on real evaluations. This limitation motivates the development of more sample-efficient multitasking frameworks, such as the one proposed in this work.

B. Generative Models

Generative models are designed to capture the underlying distribution of the training data in order to generate new, similar samples [23]. Given a dataset of observations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, the objective of a generative model is to learn a probability distribution $p_\theta(\mathbf{x})$ that approximates the true data distribution $p_{\text{data}}(\mathbf{x})$. This is often achieved by maximizing the log-likelihood function with respect to the parameters θ:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i) \tag{2}$$

A typical generative model introduces a latent variable $\mathbf{z}$, which is sampled from a prior distribution $p(\mathbf{z})$, often a Gaussian distribution. The generation process consists of two main components [24]:

1) Prior Sampling: A latent variable $\mathbf{z}$ is drawn from a prior distribution $p(\mathbf{z})$. The latent space typically has a lower dimensionality than the data space, which helps the model to efficiently learn data representations.
2) Mapping to Data Space: A function $g_\theta(\mathbf{z})$, usually parameterized by a neural network, maps the latent variable to the data space, generating a sample $\mathbf{x}$:

$$\mathbf{x} = g_\theta(\mathbf{z}) \tag{3}$$

In cases where a posterior distribution for the latent variable is approximated, the Evidence Lower Bound (ELBO) [25] is maximized instead of the direct likelihood. The ELBO is expressed as:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) \tag{4}$$

where $q_\phi(\mathbf{z}|\mathbf{x})$ approximates the posterior, and D_{KL} represents the Kullback-Leibler divergence between the approximate posterior and the prior distribution.

Generative models are widely used in tasks involving complex, high-dimensional data such as images, audio, or text [26]. By utilizing a probabilistic framework, they are capable of capturing intricate patterns and dependencies, making them suitable for diverse tasks like data synthesis, augmentation, and representation learning. Furthermore, generative models can provide uncertainty estimates, which are valuable for tasks requiring confidence measures.

In the context of evolutionary multitasking, generative models play a crucial role by providing a framework for generating diverse and high-quality individuals [27]–[29]. The latent space learned by the generative model captures the underlying structure of decision variables, which can be used to improve multitasking optimization. Specifically, the latent variables $\mathbf{z}$ can be seen as efficient encodings of the solution space, and by sampling from this space, solutions can be generated for multiple tasks with varying levels of complexity. Moreover, generative models hold great potential in ExMTOPs scenarios, where evaluating multiple tasks incurs high computational costs. By learning to map high-dimensional decision variables (e.g., $\mathbf{x} \in \mathbb{R}^{50}$) into a structured latent space, the model can generate diverse and effective candidate solutions for different tasks. Additionally, sampling from the latent space allows for simultaneous exploration of multiple tasks, thus improving the search efficiency in evolutionary multitasking.

C. Knowledge Distillation

Knowledge distillation (KD) is a model compression technique that transfers knowledge from a large, complex model, referred to as the *teacher model*, to a smaller and more efficient *student model*. This method allows the student model to retain high performance while requiring significantly fewer computational resources. It was first proposed by Hinton et al. [30] to address the need for deploying models in resource-constrained environments, where large models are impractical due to their high computational costs. The core idea of knowledge distillation is to use the output probabilities of the teacher model, also known as "soft labels," to train the student model. These soft labels contain richer information than hard labels, revealing relationships between classes that help the student model generalize better [31].

The teacher model $T(\mathbf{x}; \theta_T)$ outputs a probability distribution over the classes for an input $\mathbf{x}$, using a softmax function with a temperature parameter τ to soften the output:

$$\mathbf{p}_T = \text{softmax}\left(\frac{T(\mathbf{x}; \theta_T)}{\tau}\right) \tag{5}$$

Similarly, the student model $S(\mathbf{x}; \theta_S)$ generates a probability distribution:

$$\mathbf{p}_S = \text{softmax}\left(\frac{S(\mathbf{x}; \theta_S)}{\tau}\right) \tag{6}$$

To train the student model, the Kullback-Leibler (KL) divergence between the teacher's and student's output distributions is minimized:

$$\mathcal{L}_{KD} = \tau^2 \cdot D_{\text{KL}}\left(\mathbf{p}_T \| \mathbf{p}_S\right) \tag{7}$$

In generative models, the goal of knowledge distillation is to reduce the complexity of the generation process without compromising the ability to produce high-quality samples. The student model, distilled from the teacher model, can still generate statistically similar outputs [32]. This ensures that, despite having fewer parameters, the student model captures the key patterns and distributions necessary for high-quality generation, making it a promising approach for addressing ExMTOPs. Moreover, knowledge distillation could significantly reduce inference cost [33], enabling faster solution generation during optimization.

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2025.3617343

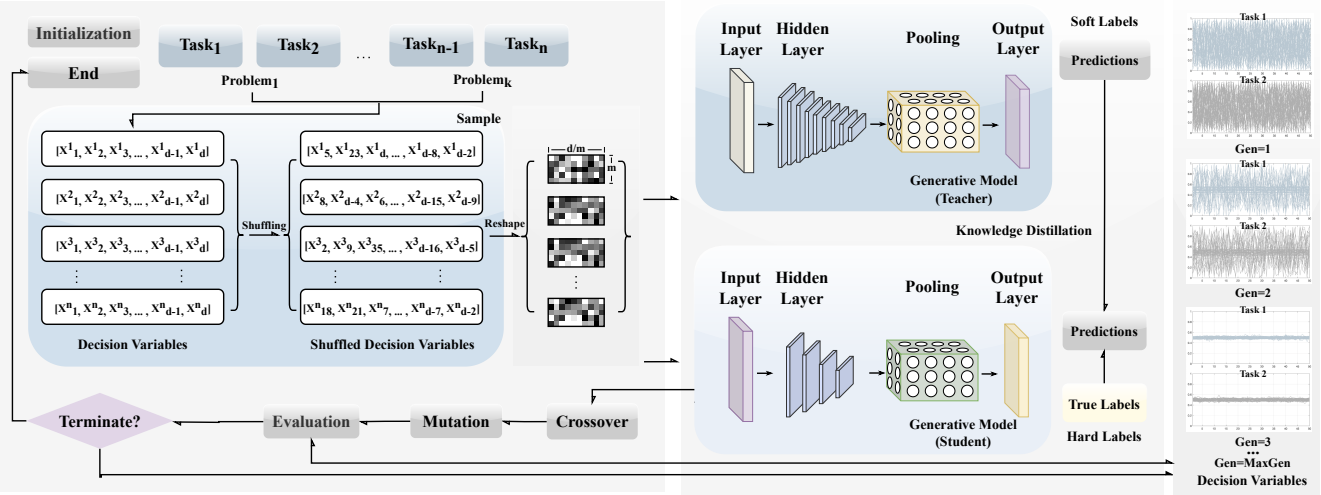


Fig. 1: MFEA-SSG algorithm architecture diagram.

III. Single-Step Generative Model for ExMTOPs

In this section, we first introduce the proposed meta-learning-inspired ExMTOPs algorithm based on a single-step generative model, detailing the role of meta-learning in enhancing algorithm performance and explaining the algorithm's workflow. We then discuss the advantages of the single-step generative model. Finally, we provide a theoretical analysis showing how knowledge distillation significantly reduces the computational complexity of the student model compared to the teacher model.

A. MFEA-SSG Algorithm

Existing surrogate-based methods face significant challenges in high-dimensional multitask optimization. They often fail to generalize and adapt across tasks. To address these challenges, we introduce the MFEA-SSG algorithm. Instead of approximating objective functions, MFEA-SSG leverages a generative model to learn the distribution of high-quality solutions, thereby reducing reliance on expensive evaluations. Inspired by meta-learning, we introduce a variable-shuffling and reshaping strategy to construct task-agnostic representations. This is achieved by treating each random shuffle as a pseudo-task, thereby training the model on a distribution of tasks to compel it to capture transferable structures across tasks and improve generalization. This is achieved by treating each random shuffle as a pseudo-task, thereby training the model on a distribution of tasks to compel it to capture transferable structures across tasks and improve generalization. These structural patterns are not properties of the fitness landscape itself, but rather the statistical properties and inter-variable dependencies characteristic of high-quality solutions. For instance, the model learns the typical value distributions for individual variables and, more importantly, the correlations between them (e.g., that a high value for one variable often accompanies a low value for another in elite solutions). The meta-learning-inspired shuffling compels the model to learn these dependencies in a generalized, dimension-agnostic manner. To further reduce inference cost, knowledge distillation is applied to compress the complex generative model into a simplified student model capable of single-step generation. This generative model undergoes a process of progressive refinement, where it adaptively learns from high-performing solutions discovered during the evolutionary search.

The overall architecture of MFEA-SSG is illustrated in Fig. 1. The framework operates as a feedback loop between the traditional evolutionary search process and our generative transfer mechanism. The process begins with the evolutionary algorithm exploring the search space. To facilitate knowledge transfer and prepare data for the generative model, a crucial preprocessing pipeline is applied. First, to handle tasks with potentially different search space boundaries, all decision variables are normalized to a unified range of [0, 1]. High-performing individuals are then continuously sampled from the evolving population, and their dimensions are shuffled and reshaped into image-like representations. This data is used to progressively refine a powerful, multi-step diffusion model (the Teacher), from which a lightweight, single-step Student model is distilled. This efficient Student model then acts as the core of the generative framework, producing new, high-quality individuals that are fed back into the evolutionary loop to guide subsequent generations.

The detailed procedure is formalized in Algorithm 1. After initializing a shared population P (line 1), the main evolutionary loop (lines 2-22) iteratively generates and evaluates new individuals. The offspring production mechanism (lines 4-16) is conditioned by the evolutionary progress. In the early stages ($gen \leq MaxGen$), the algorithm primarily leverages the generative model G for exploration, producing a novel candidate solution which is then enhanced by mutation (lines 6-9). In later stages, the framework transitions to using conventional genetic crossover for exploitation (lines 11-14). After the new offspring population is evaluated and used to update the main population P (line 17), the framework's core refinement process is triggered periodically. If the generation count is a multiple of the refinement frequency τ, the generative model G is progressively refined using the latest high-quality individuals (lines 18-20). Finally, the generation

Algorithm 1 MFEA-SSG: Multifactorial Evolutionary Algorithm-Single Step Generative Model

Require: Population size N, number of tasks T, Random Mating Probability RMP, Crossover parameter MuC, Mutation parameter MuM, generative model G, max generation for generative phase $MaxGen$, refinement frequency τ

Ensure: Optimized population for all tasks

1: Initialize population P; $gen \leftarrow 0$
2: **while** termination condition not met **do**
3: $Offspring \leftarrow \emptyset$
4: **for** each pair of parents (p_1, p_2) selected from P **do**
5: **if** $gen \leq MaxGen$ **and** (Same task or rand(0,1) $<$ RMP)) **then**
6: Generate decisions Dec from noise Z using G
7: Apply mutation: $Dec' \leftarrow$ GA_Mutation(Dec, MuM)
8: Create one offspring o_1 with decisions Dec'
9: Assign a parent's task factor to o_1 and add to $Offspring$
10: **else**
11: Apply crossover: $[Dec'_1, Dec'_2] \leftarrow$ GA_Crossover$(p_1.Dec, p_2.Dec, MuC)$
12: Create two offspring o_1, o_2 with decisions Dec'_1, Dec'_2
13: Assign parents' task factors to o_1, o_2 and add to $Offspring$
14: **end if**
15: **end for**
16: Evaluate $Offspring$, and update population P by selecting from $P \cup Offspring$
17: **if** $mod(gen, \tau) == 0$ **then**
18: Progressively refine generative model G
19: **end if**
20: $gen \leftarrow gen + 1$
21: **end while**
22: Output optimized population

counter is incremented (line 21), and the loop continues until termination.

B. Meta-Learning Inspired Knowledge Distillation for a Simplified Generative Model

We adopt a denoising diffusion probabilistic model to learn the distribution of promising solutions in ExMTOPs. The full training procedure is summarized in Algorithm 2, and its key steps are described as follows: The training data consists of individuals from different tasks, each originally represented as a 1×50 vector. Since the order of decision variables (e.g., $x_1, x_2, ..., x_{50}$) is arbitrary and does not reflect spatial semantics, reshaping them directly into 2D grids may introduce misleading spatial relationships based on artificial adjacencies. To avoid this, we first apply random shuffling before reshaping. This shuffling is applied independently to each individual sample to maximize the diversity of the recomposed inputs, ensuring the model does not learn spurious

Algorithm 2 Training Process for Generative Model

Require: Training data I sampled from the evolutionary process, generative model G, noise schedule $\{\beta_t\}_{t=1}^N$, number of noise steps N, learning rate α, number of epochs E, mini-batch size B

Ensure: Trained generative model G (Teacher)

1: **for** epoch $e = 1$ to E **do**
2: Shuffle the training data
3: **for** each mini-batch of individuals I of size B **do**
4: **for** each individual i in I **do**
5: Randomly shuffle dimensions of individual i # *Break spatial dependencies*
6: Reshape i into a 5×10 image
7: **end for**
8: Sample random noise $\epsilon \sim \mathcal{N}(0, 1)$
9: Randomly select noise step $t \sim$ Uniform$(1, N)$
10: Apply noise to reshaped images: $X_t = \sqrt{\alpha_t} X + \sqrt{1 - \alpha_t}\epsilon$
11: Forward pass: Generate prediction from model G:
12: $G(X_t, t) \rightarrow \hat{\epsilon}_G$
13: Compute loss: $L = \frac{1}{B} \sum_{i=1}^{B} \|\hat{\epsilon}_{G_i} - \epsilon_i\|^2$
14: Update G using Adam optimizer: $\theta_G \leftarrow \theta_G - \alpha \nabla_{\theta_G} L$
15: **end for**
16: **end for**
17: **Output:** Trained generative model G

dependencies based on the arbitrary ordering of variables. This process is crucial as it encourages the model to learn truly task-agnostic representations, significantly improving its generalization capabilities across heterogeneous problems (line 5), and the resulting vectors are reshaped into 5×10 image-like inputs (line 6). In each epoch, noise is added to the data according to a forward diffusion schedule $\{\beta_t\}_{t=1}^T$ (lines 9–10). The generative model G predicts the noise component $\hat{\epsilon}_G$ and is updated using mean squared error loss with the Adam optimizer (lines 13–14). After sufficient training epochs, the model G is capable of generating high-quality individuals (line 17).

Formally, at each diffusion timestep t, a noisy sample x_t is generated as:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{8}$$

where ϵ is a noise sample drawn from a standard normal distribution, with the same dimensions as the input x_0, $\alpha_t = \prod_{s=1}^{t}(1 - \beta_s)$. The diffusion model ϵ_θ learns to predict ϵ by minimizing:

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{x_0, t, \epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]. \tag{9}$$

The model architecture follows a U-Net-style encoder–decoder with residual blocks, skip connections, and self-attention layers. Sinusoidal embeddings are used to encode timestep t.

While the multi-step diffusion process is powerful for learning complex solution distributions, its high inference cost makes it impractical for direct use within the evolutionary

loop of an ExMTOP. To overcome this critical efficiency bottleneck, we introduce a simplified student model ϵ_ϕ via knowledge distillation. The objective is to compress the end-to-end generative capability of the powerful teacher model (ϵ_θ) into the lightweight, single-step student. This allows us to retain the high-fidelity generation learned by the full diffusion model, while enabling the rapid sampling required for an efficient evolutionary search. In each distillation iteration (Algorithm 3), we first sample a clean input x_0 and randomly select a timestep t. The corresponding noisy version x_t is generated, and the teacher model outputs the ground truth noise prediction $\epsilon_\theta(x_t, t)$. The student model ϵ_ϕ then takes x_t and t as input and produces its own prediction. The student is trained by minimizing the discrepancy between its output and the teacher's target:

$$\mathcal{L}_{\text{KD}}(\phi) = \mathbb{E}_{x_0,t}\left[\left\|\epsilon_\theta(x_t,t) - \epsilon_\phi(x_t,t)\right\|^2\right]. \tag{10}$$

Algorithm 3 Knowledge Distillation for Single-Step Generative Student Model

Require: Trained teacher model T, initialized student model S, training inputs x_0 of size 5×10, noise schedule $\{\beta_t\}_{t=1}^N$, number of noise steps N, learning rate α, number of epochs E, mini-batch size B

Ensure: Trained student model S for single-step inference

Initialize student model S with parameters θ_S
for epoch $e = 1$ to E **do**
  Shuffle training data
  for each mini-batch $\{x_0^{(i)}\}_{i=1}^B$ **do**
    Sample noise $\epsilon \sim \mathcal{N}(0, I)$
    Sample timestep $t \sim \text{Uniform}(1, N)$
    Compute noisy input: $x_t = \sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\epsilon$
    Get predictions: $\hat{\epsilon}_T = T(x_t, t), \quad \hat{\epsilon}_S = S(x_t, t)$
    Compute loss: $L = \frac{1}{B}\sum_{i=1}^B \|\hat{\epsilon}_{S_i} - \hat{\epsilon}_{T_i}\|^2$
    Update θ_S via gradient descent: $\theta_S \leftarrow \theta_S - \alpha\nabla_{\theta_S} L$
  end for
end for
Output: Trained student model S for single-step generation

Unlike the teacher model, which must iteratively denoise over T steps during inference, the student model supports single-step generation: it directly maps a noisy input x_t to a high-quality denoised sample in one forward pass. This enables significant speedup in sample generation while preserving solution quality.

C. Complexity analysis between student and teacher models

In this section, we provide a theoretical analysis of the computational complexity between the student and teacher models based on their respective architectures. The complexity is measured in terms of floating-point operations (FLOPs) [34], representing the total number of multiplications and additions required during inference.

The computational complexity of a 2D convolutional layer can be expressed as:

$$\text{FLOPs}_{\text{Conv}} = 2 \times H \times W \times C_{\text{in}} \times C_{\text{out}} \times K_H \times K_W \tag{11}$$

Where H and W denote the height and width of the input feature map, C_{in} and $Cout$ are the numbers of input and output channels, and K_H and K_W are the height and width of the convolutional kernel, respectively.

A residual block typically consists of two convolutional layers, each followed by a non-linearity. The FLOPs for a residual block can thus be approximated as twice the FLOPs of a single convolutional layer:

$$\text{FLOPs}_{\text{ResBlock}} = 2 \times \text{FLOPs}_{\text{Conv}} \tag{12}$$

Attention mechanisms involve matrix multiplications, which can significantly increase computational complexity, especially with large feature maps and multiple attention heads. For simplicity, we approximate the FLOPs of an attention block as:

$$\text{FLOPs}_{\text{Attention}} \approx 2 \times (H \times W \times C)^2 \tag{13}$$

Where C is the number of channels.

The student model architecture includes an input layer (a standard convolutional layer), an encoder with one residual block and one downsampling convolutional layer, a bridge consisting of one residual block, a decoder with one transposed convolutional layer followed by a residual block, and an output convolutional layer. The total FLOPs for the student model can be expressed as the sum of the FLOPs of these components:

$$\begin{aligned}\text{FLOPs}_{\text{Student}} = {} & \text{FLOPs}_{\text{Conv}_{\text{m}}} + \text{FLOPs}_{\text{ResBlock}_1} \\ & + \text{FLOPs}_{\text{Downsample}} + \text{FLOPs}_{\text{ResBlock}_2} \\ & + \text{FLOPs}_{\text{Upsample}} + \text{FLOPs}_{\text{ResBlock}_3} \\ & + \text{FLOPs}_{\text{Conv}_{\text{out}}}\end{aligned} \tag{14}$$

In contrast, the teacher model architecture is significantly more complex. It comprises a standard convolutional layer, an encoder with multiple residual blocks interleaved with attention blocks and followed by two downsampling convolutional layers, a bridge with multiple residual blocks incorporating attention mechanisms, a decoder with multiple transposed convolutional layers accompanied by residual and attention blocks, and an output convolutional layer. The total FLOPs for the teacher model can be expressed as:

$$\begin{aligned}\text{FLOPs}_{\text{Teacher}} = {} & \text{FLOPs}_{\text{Conv}_{\text{in}}} + \sum_{i=1}^{n_1}(\text{FLOPs}_{\text{ResBlock}_i} \\ & + \text{FLOPs}_{\text{Attention}_i}) + \text{FLOPs}_{\text{Downsam}} \\ & + \sum_{j=1}^{n_2}(\text{FLOPs}_{\text{ResBlock}_j} + \text{FLOPs}_{\text{Attention}_j}) \\ & + \text{FLOPs}_{\text{Downsample}_4} + \text{FLOPs}_{\text{Bridge}} \\ & + \text{FLOPs}_{\text{Decoder}} + \text{FLOPs}_{\text{Conv}_{\text{out}}}\end{aligned} \tag{15}$$

where n_1 and n_2 represent the number of residual and attention blocks in different stages of the encoder.

To quantify the computational efficiency gained by using the student model, we define the complexity ratio as the FLOPs of the teacher model divided by that of the student model:

$$\text{Complexity Ratio} = \frac{\text{FLOPs}_{\text{Teacher}}}{\text{FLOPs}_{\text{Student}}} \tag{16}$$

Due to the teacher model's deeper architecture, larger channel sizes, and extensive use of attention and residual blocks, this ratio is expected to be significantly greater than 1. In contrast, the student model adopts a streamlined structure with fewer layers and no attention mechanisms. Assuming comparable input dimensions and kernel configurations, the student model reduces the overall inference cost by approximately 10 to 20 times, enabling much faster generation while preserving solution quality.

To provide a clearer picture of the computational benefits, we estimate the FLOPs of both the teacher and student models under a representative configuration. Specifically, we consider an input of size $5\times10\times1$, with convolutional layers using 3×3 kernels and channel sizes ranging from 64 to 256. The results are summarized in Table I.

TABLE I: COMPARISON OF TEACHER AND STUDENT MODEL COMPLEXITY.

Model	Residual Blocks	Attention Blocks	Total FLOPs (Millions)	Relative Cost
Teacher Model	10	4	1920	18x
Student Model	3	0	106	1x

As shown, the teacher model involves significantly more convolutional operations, deeper residual paths, and multiple attention blocks, leading to substantially higher FLOPs. In contrast, the student model adopts a streamlined architecture without attention layers and with fewer residual blocks. In this typical case, the total inference cost of the student model is reduced by approximately $18\times$, aligning with our theoretical expectations.

IV. Experimental Studies

In this section, we comprehensively validate the superiority of the proposed single-step generative model-based MFEA algorithm from four perspectives. First, we compare its performance with recently introduced high-performing EMT general algorithms. Then, we conduct an in-depth analysis and comparison with other EMT algorithms designed for expensive scenarios. In the third part, we evaluate the improvements in time complexity of the distilled model and visualize the decision variables in comparison with other algorithms. Finally, we explore how meta-learning can be employed to further enhance the algorithm's performance on tasks where it underperforms, and we conduct an ablation study to assess the impact of the meta-learning approach on our algorithm.

A. Experimental Configuration

The experimental parameters are organized into two main categories. For the evolutionary and experimental setup, we followed the standard settings commonly adopted in related works to ensure a fair comparison. The population size for all algorithms was set to 100, with a random mating probability of 0.3, a polynomial mutation value of 5, and a simulated binary crossover value of 2. Based on the recommendations in [6], the maximum number of function evaluations (MaxFEs) was fixed at 400. Notably, these general parameters were not specifically tuned for our proposed algorithm but were adopted from the standard settings in the field. For the generative model, we also employed standard and robust settings from the machine learning literature. The model was trained for 5 epochs with a batch size of 512. We used the Adam optimizer with a learning rate of 0.0005, a gradient decay factor (β_1) of 0.9, and a squared gradient decay factor (β_2) of 0.9999. The number of input channels was 1. These choices were made to ensure stable and reliable training without problem-specific hyperparameter tuning. All experiments, except for those involving expensive algorithms, were conducted on the MTO-platform [35].

B. Comparison with State-of-the-Art EMT Algorithms

The proposed MFEA-SSG algorithm is evaluated using the Competition on Evolutionary Computation - 2017 Multitask Single Objective benchmark functions (CEC-2017-MTSO). The CEC-2017-MTSO benchmark comprises the following categories: CI-HS, CI-MS, CI-LS, PI-HS, PI-MS, PI-LS, NI-HS, NI-MS, and NI-LS. In the experimental result tables, we refer to each problem category as P1 through P9, respectively. More specifically, each category contains two different types of optimization tasks: CI represents complete intersection, PI stands for partial intersection, and NI denotes no intersection. Additionally, HS, MS, and LS indicate high, medium, and low similarity, respectively. This diverse set of problem configurations enables a more comprehensive evaluation of the proposed algorithm's performance. To evaluate the significance of performance differences, we conducted a Wilcoxon rank-sum test at a significance level of 0.05. The statistical outcomes are presented in the last row of the table using the symbols +/-/=, where "+" indicates that the MFEA-SSG algorithm outperforms others, "-" indicates that MFEA-SSG performs inferiorly to the compared algorithm, and "=" denotes similar performance. Additionally, the "Std" row displays the standard deviation from multiple runs, representing the variability in performance.

Table II and Fig. 2 presents a comparative analysis of the proposed MFEA-SSG algorithm against several state-of-the-art general EMT algorithms, including AT-MFEA [36], MFEA [22], MFEA-II [37], ASCMFDE [38], BLKT-DE [39], MTEA-SaO [40], MTES [41] and MTES-KG [42]. Performance is measured across a variety of tasks, using the best objective values achieved for each task. The results clearly show that MFEA-SSG consistently outperforms the other algorithms in most tasks, as indicated by the lower objective values for MFEA-SSG (highlighted in grey). The convergence trends in Fig. 2 further illustrate the superior performance of MFEA-SSG across different tasks (P1 to P9). Each plot shows the logarithmic objective values over evaluations, highlighting the faster and more stable convergence of MFEA-SSG compared

TABLE II: COMPARISON OF MFEA-SSG WITH STATE-OF-THE-ART MULTITASK OPTIMIZATION ALGORITHMS.

Category	MFEA-SSG	AT-MFEA	MFEA	MFEA-II	ASCMFDE	BLKT-DE	MTEA-SaO	MTES	MTES-KG
P1-T1	**9.8316e-01**	2.2455e+01 -	2.4592e+01 -	2.3001e+01 -	2.6046e+01 -	2.5101e+01 -	2.1988e+01 -	4.0484e+00 -	4.3475e+00 -
Std	(5.94e-02)	(1.76e+00)	(1.55e+00)	(1.64e+00)	(2.37e+00)	(2.67e+00)	(2.84e+00)	(2.98e-01)	(1.08e+00)
P1-T2	**4.3386e+02**	2.2244e+04 -	2.3913e+04 -	2.2290e+04 -	2.5108e+04 -	2.4590e+04 -	2.0931e+04 -	2.0582e+03 -	3.8228e+03 -
Std	(2.18e+01)	(2.18e+03)	(1.64e+03)	(1.69e+03)	(1.98e+03)	(2.42e+03)	(2.97e+03)	(1.15e+03)	(1.07e+03)
P2-T1	**4.2517e+00**	2.1287e+01 -	2.1299e+01 -	2.1309e+01 -	2.1300e+01 -	2.1309e+01 -	2.1288e+01 -	1.7558e+01 -	1.7652e+01 -
Std	(1.54e-01)	(4.87e-02)	(4.41e-02)	(4.41e-02)	(4.53e-02)	(4.65e-02)	(6.50e-02)	(2.99e-01)	(7.91e-01)
P2-T2	**4.3186e+02**	2.1388e+04 -	2.4149e+04 -	2.1197e+04 -	2.5404e+04 -	2.6418e+04 -	2.2108e+04 -	1.6751e+03 -	3.8111e+03 -
Std	(2.89e+01)	(1.35e+03)	(1.76e+03)	(2.17e+03)	(1.96e+03)	(2.59e+03)	(3.01e+03)	(8.49e+02)	(8.19e+02)
P3-T1	**2.1384e+01**	2.1374e+01 =	2.1353e+01 =	2.1371e+01 =	2.1378e+01 =	2.1370e+01 =	2.1380e+01 =	2.1355e+01 =	2.1378e+01 =
Std	(3.72e-02)	(3.92e-02)	(6.74e-02)	(4.45e-02)	(5.15e-02)	(5.22e-02)	(4.95e-02)	(4.58e-02)	(4.61e-02)
P3-T2	1.6772e+04	1.5557e+04 +	1.6013e+04 +	1.5685e+04 +	1.6848e+04 =	1.6681e+04 =	1.6145e+04 +	**1.0785e+04** +	1.7423e+04 -
Std	(7.59e+02)	(6.35e+02)	(4.62e+02)	(5.89e+02)	(3.97e+02)	(6.71e+02)	(4.81e+02)	(3.06e+03)	(4.79e+02)
P4-T1	**4.3073e+02**	2.2530e+04 -	2.2689e+04 -	2.1576e+04 -	2.5238e+04 -	2.5116e+04 -	2.1739e+04 -	3.4215e+03 -	4.1781e+03 -
Std	(2.56e+01)	(1.64e+03)	(2.06e+03)	(1.67e+03)	(2.16e+03)	(2.32e+03)	(2.36e+03)	(3.16e+02)	(7.41e+02)
P4-T2	**4.8321e+03**	8.6357e+04 -	9.6581e+04 -	8.8575e+04 -	1.0371e+05 -	1.0224e+05 -	8.8439e+04 -	1.6265e+04 -	2.3468e+04 -
Std	(1.38e+02)	(6.64e+03)	(6.77e+03)	(6.44e+03)	(8.34e+03)	(9.89e+03)	(9.08e+03)	(4.65e+03)	(4.60e+03)
P5-T1	**4.7176e+00**	2.1285e+01 -	2.1316e+01 -	2.1278e+01 -	2.1311e+01 -	2.1318e+01 -	2.1272e+01 -	1.7406e+01 -	1.7889e+01 -
Std	(2.32e-01)	(5.61e-02)	(3.96e-02)	(8.45e-02)	(9.09e-02)	(4.22e-02)	(1.06e-01)	(3.03e-01)	(1.08e+00)
P5-T2	**5.7168e+03**	2.1935e+09 -	2.4308e+09 -	2.1107e+09 -	2.8790e+09 -	3.1143e+09 -	2.4032e+09 -	3.0833e+07 -	8.1004e+07 -
Std	(1.41e+03)	(2.85e+08)	(3.23e+08)	(2.83e+08)	(3.68e+08)	(4.23e+08)	(4.19e+08)	(5.68e+07)	(5.52e+07)
P6-T1	**4.2738e+00**	2.1295e+01 -	2.1307e+01 -	2.1281e+01 -	2.1309e+01 -	2.1303e+01 -	2.1275e+01 -	1.7303e+01 -	1.7803e+01 -
Std	(2.53e-01)	(5.12e-02)	(5.68e-02)	(6.91e-02)	(5.29e-02)	(7.11e-02)	(1.21e-01)	(4.11e-01)	(8.96e-01)
P6-T2	**4.5765e+00**	3.3839e+01 -	3.3513e+01 -	3.4005e+01 -	3.5283e+01 -	3.5066e+01 -	3.3891e+01 -	1.6645e+01 -	2.4093e+01 -
Std	(5.56e-01)	(1.36e+00)	(1.12e+00)	(9.97e-01)	(1.34e+00)	(1.76e+00)	(1.15e+00)	(3.05e+00)	(3.41e+00)
P7-T1	**6.4710e+03**	2.2855e+09 -	2.4205e+09 -	2.2901e+09 -	3.1096e+09 -	2.9688e+09 -	2.4330e+09 -	5.0086e+07 -	7.5983e+07 -
Std	(2.15e+03)	(2.52e+08)	(3.36e+08)	(3.88e+08)	(5.02e+08)	(5.16e+08)	(5.08e+08)	(8.29e+06)	(3.41e+07)
P7-T2	**4.3991e+02**	2.2114e+04 -	2.3403e+04 -	2.2231e+04 -	2.5332e+04 -	2.4994e+04 -	2.2178e+04 -	2.4530e+03 -	4.2430e+03 -
Std	(2.32e+01)	(1.69e+03)	(1.64e+03)	(1.46e+03)	(2.61e+03)	(2.38e+03)	(2.75e+03)	(1.46e+03)	(1.01e+03)
P8-T1	**1.0023e+00**	2.3137e+01 -	2.5988e+01 -	2.2459e+01 -	2.5115e+01 -	2.5434e+01 -	2.1844e+01 -	4.6140e+00 -	5.7946e+00 -
Std	(3.29e-02)	(1.17e+00)	(1.55e+00)	(1.56e+00)	(1.75e+00)	(2.47e+00)	(2.67e+00)	(4.01e-01)	(1.11e+00)
P8-T2	**1.0393e+01**	7.4199e+01 -	7.4971e+01 -	7.4124e+01 -	7.5234e+01 -	7.6198e+01 -	7.4480e+01 -	3.5851e+01 -	4.9778e+01 -
Std	(8.16e-01)	(1.40e+00)	(1.90e+00)	(3.27e+00)	(2.21e+00)	(1.79e+00)	(2.30e+00)	(6.98e+00)	(4.05e+00)
P9-T1	**4.4079e+02**	2.1446e+04 -	2.4002e+04 -	2.1295e+04 -	2.5309e+04 -	2.6107e+04 -	2.2140e+04 -	3.4731e+03 -	4.3799e+03 -
Std	(2.10e+01)	(1.55e+03)	(2.02e+03)	(2.04e+03)	(2.12e+03)	(2.56e+03)	(1.93e+03)	(3.47e+02)	(1.17e+03)
P9-T2	1.6828e+04	1.5897e+04 +	1.6268e+04 +	1.5762e+04 +	1.6700e+04 =	1.6641e+04 =	1.6108e+04 +	**1.1215e+04** +	1.7496e+04 -
Std	(8.90e+02)	(3.73e+02)	(5.42e+02)	(5.79e+02)	(4.05e+02)	(4.04e+02)	(3.71e+02)	(3.29e+03)	(4.70e+02)
+ / - / =	Base	2 / 15 / 1	2 / 15 / 1	2 / 15 / 1	0 / 15 / 3	0 / 15 / 3	2 / 15 / 1	2 / 16 / 0	0 / 17 / 1

to the other algorithms. In most tasks, MFEA-SSG achieves lower objective values with fewer evaluations, emphasizing its efficiency in reaching high-quality solutions with limited computational resources. The rapid convergence trend observed for MFEA-SSG, especially in tasks P1, P2, P4, and P7, underscores its advantage in ExMTOPs scenarios where the number of evaluations is restricted.

Most general-purpose EMT algorithms rely heavily on frequent objective evaluations to ensure effective knowledge transfer and fitness estimation. However, in ExMTOPs with stringent evaluation limits, this reliance results in substantial computational costs. These methods typically require 50,000 to 100,000 evaluations to achieve competitive performance, which is infeasible in many real-world applications. Moreover, when inter-task similarity is low, knowledge transfer based solely on this similarity can lead to negative transfer—injecting misleading information that disrupts the search process. This challenge becomes more severe in heterogeneous multitask settings, where the diversity of task features and decision spaces undermines the unified optimization strategy adopted by traditional algorithms.

The superior performance of MFEA-SSG across diverse tasks demonstrates the effectiveness of integrating generative modeling with ExMTOPs. By learning the distribution of promising decision regions during pre-training, the generative model reduces the dependency on real evaluations, thus accelerating convergence under tight evaluation budgets. This is reflected in the significantly lower number of evaluations required to reach competitive performance compared to baseline algorithms. Additionally, the model's robustness in heterogeneous tasks can be attributed to its meta-learning-inspired representation strategy. The random shuffling and reshaping of decision variables disrupt task-specific dependencies and mitigate overfitting, enabling the model to generalize better across tasks with varying dimensional structures. This contributes to consistently strong performance across both homogeneous and heterogeneous benchmarks.

C. Comparison with State-of-the-Art ExMTOPs Algorithms

This section presents a comparative analysis of MFEA-SSG against other state-of-the-art ExMTOPs algorithms, including several powerful Surrogate-Assisted Evolutionary Algorithms (SAEAs), to benchmark its performance in evaluation-constrained scenarios. The selected baselines include G-MFEA [1], a foundational algorithm for heterogeneous multitasking; standard Bayesian Optimization (BO) [43]; and two highly competitive SAEAs, EEI-BO [44] and SELF [6].

EEI-BO is an advanced BO framework that utilizes a Gaussian Process (GP) surrogate model. Its core innovation is the Evolutionary Expected Improvement (EEI) infill criterion,

which integrates the population distribution from a Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to guide the search more effectively toward promising regions and reduce wasted evaluations. SELF is a sophisticated surrogate-assisted framework designed specifically for ExMTOPs, featuring a two-phase knowledge transfer mechanism. In its global phase, it employs a Multitask Gaussian Process (MTGP) model to capture and share landscape knowledge across all tasks. In its local phase, it uses task-specific BO to further refine the best-found solutions. Both algorithms represent the state-of-the-art in leveraging surrogate models for expensive optimization.

The comparative results are presented in Table III. In almost all subtasks from P1 to P9, MFEA-SSG consistently and significantly outperforms all competitors, including the strong SAEA baselines. This demonstrates the superiority of our generative approach in ExMTOPs settings. The performance advantage can be attributed to a fundamental difference in methodology: while SAEAs like SELF and EEI-BO focus on approximating the complex fitness landscape, MFEA-SSG circumvents this by directly learning the distribution of high-quality solutions. This strategy appears to be more direct and sample-efficient, enabling the generation of promising candidates without needing to model the entire solution-to-fitness mapping, which is a particularly challenging task under a severely limited evaluation budget. It is important to note that the SELF algorithm is not applicable to heterogeneous problems; therefore, results for the tasks corresponding to P6 and P8 are not provided.

TABLE III: COMPARISON OF MFEA-SSG WITH STATE-OF-THE-ART EXPENSIVE MULTITASK OPTIMIZATION ALGORITHMS.

Category	MFEA-SSG	G-MFEA	BO	EEI-BO	SELF
P1-T1	**9.8316e-01**	2.7122e+01-	8.3834e+00-	1.4848e+00-	1.1617e+00-
Std	(5.94e-02)	(2.07e+00)	(5.38e+00)	(8.08e-02)	(4.41e-02)
P1-T2	**4.3386e+02**	2.6117e+04-	7.7792e+03-	9.2464e+02-	6.0959e+02-
Std	(2.18e+01)	(1.70e+03)	(2.61e+03)	(7.99e+01)	(6.01e+01)
P2-T1	**4.2517e+00**	2.1282e+01-	1.9453e+01-	1.0172e+01-	1.2413e+01-
Std	(1.54e-01)	(4.86e-02)	(1.21e+00)	(5.35e-01)	(1.05e+00)
P2-T2	**4.3186e+02**	2.6514e+04-	8.5914e+03-	8.9341e+02-	1.0482e+03-
Std	(2.89e+01)	(1.78e+03)	(2.56e+03)	(8.34e+01)	(1.86e+02)
P3-T1	2.1384e+01	2.1388e+01=	2.1460e+01=	2.1383e+01=	**2.1377e+01=**
Std	(3.72e-02)	(4.39e-02)	(6.28e-02)	(5.14e-02)	(5.10e-02)
P3-T2	1.6772e+04	1.6680e+04=	1.7831e+04=	1.7157e+04=	**1.4868e+04+**
Std	(7.59e+02)	(5.17e+02)	(4.92e+02)	(4.35e+02)	(1.53e+03)
P4-T1	**4.3073e+02**	2.6866e+04-	7.7129e+03-	1.0409e+03-	8.6435e+02-
Std	(2.56e+01)	(1.55e+03)	(3.32e+03)	(1.12e+02)	(1.48e+02)
P4-T2	4.8321e+03	1.0945e+05-	3.6715e+04-	2.4791e+03+	**1.4998e+03+**
Std	(1.38e+02)	(7.24e+03)	(1.02e+04)	(4.41e+02)	(5.47e+02)
P5-T1	**4.7176e+00**	2.1290e+01-	1.8897e+01-	1.0647e+01-	1.4486e+01-
Std	(2.32e-01)	(4.44e-02)	(1.08e+00)	(7.12e-01)	(9.46e-01)
P5-T2	**5.7168e+03**	2.7410e+09-	4.1016e+08-	2.1020e+06-	6.0778e+06-
Std	(1.41e+03)	(4.15e+08)	(1.82e+08)	(1.21e+06)	(2.59e+06)
P6-T1	**4.2738e+00**	2.1310e+01-	1.9495e+01-	1.1351e+01-	-
Std	(2.53e-01)	(5.92e-02)	(1.42e+00)	(9.02e-01)	-
P6-T2	**4.5765e+00**	3.4487e+01-	1.8608e+01-	9.4203e+00-	-
Std	(5.56e-01)	(1.27e+00)	(5.88e+00)	(1.17e+00)	-
P7-T1	**6.4710e+03**	3.0060e+09-	4.7674e+08-	3.2482e+06-	1.1460e+06-
Std	(2.15e+03)	(3.18e+08)	(1.98e+08)	(1.13e+06)	(4.70e+05)
P7-T2	**4.3991e+02**	2.7077e+04-	8.2691e+03-	9.8682e+02-	7.8660e+02-
Std	(2.32e+01)	(1.89e+03)	(3.25e+03)	(1.01e+02)	(1.09e+02)
P8-T1	**1.0023e+00**	2.7494e+01-	9.4661e+00-	1.5881e+00-	-
Std	(3.29e-02)	(1.50e+00)	(3.83e+00)	(1.19e-01)	-
P8-T2	**1.0393e+01**	7.4884e+01-	1.9129e+01-	8.6156e+00-	-
Std	(8.16e-01)	(2.46e+00)	(4.96e+00)	(6.95e-01)	-
P9-T1	**4.4079e+02**	2.5581e+04-	7.7622e+03-	1.0811e+03-	1.2622e+03-
Std	(2.10e+01)	(2.05e+03)	(2.40e+03)	(1.23e+02)	(3.35e+02)
P9-T2	1.6828e+04	1.6732e+04=	1.7881e+04=	1.7186e+04=	**1.6423e+04=**
Std	(8.90e+02)	(4.31e+02)	(5.90e+02)	(5.09e+02)	(5.27e+02)
+ / - / =	Base	0 / 15 / 3	0 / 15 / 3	1 / 14 / 3	2 / 9 / 3

TABLE IV: RUNTIME COMPARISON OF MFEA-SSG WITH STATE-OF-THE-ART EXPENSIVE MULTITASK OPTIMIZATION ALGORITHMS.

Task	Teacher	Student	G-MFEA	EEI-BO	SELF	BO
P1	$6.2999e+01$-	$2.5415e+00$	$8.1482e-01$+	$9.2613e+02$-	$2.7867e+02$-	$9.0830e+00$-
P2	$5.9393e+01$-	$4.6921e-01$	$9.4825e-01$-	$9.4397e+02$-	$2.7619e+02$-	$9.1000e+00$-
P3	$5.8449e+01$-	$1.3563e+00$	$3.5764e-01$+	$1.0618e+03$-	$7.5347e+01$-	$9.2380e+00$-
P4	$5.5391e+01$-	$3.2774e-01$	$5.8281e-01$-	$9.7637e+02$-	$2.8518e+02$-	$9.0320e+00$-
P5	$5.3845e+01$-	$1.6383e+00$	$2.8441e-01$+	$9.8796e+02$-	NaN	$9.0960e+00$-
P6	$5.0536e+01$-	$2.9130e-01$	$4.6024e-01$-	$8.5365e+02$-	$2.8601e+02$-	$8.9770e+00$-
P7	$5.2808e+01$-	$2.6703e-01$	$3.4758e-01$-	$9.7753e+02$-	$2.8691e+02$-	$9.0620e+00$-
P8	$5.2613e+01$-	$3.4303e-01$	$5.6306e-01$-	$8.4876e+02$-	NaN	$8.9620e+00$-
P9	$5.3309e+01$-	$2.2326e-01$	$3.7431e-01$-	$9.7561e+02$-	$3.0147e+02$-	$9.2090e+00$-
+ / - / =	0 / 9 / 0	Base	3 / 6 / 0	0 / 9 / 0	0 / 9 / 0	0 / 9 / 0

D. Comparison with General State-of-the-Art Evolutionary Multitasking Algorithms

As observed in our experiments, most surrogate-based ExMTOPs algorithms suffer from high runtime and slow convergence, particularly in heterogeneous task scenarios. These methods often fall into local optima due to their reliance on locally fitted models, and struggle to maintain accuracy across diverse, high-dimensional search spaces. In practice, their performance tends to degrade when inter-task similarity is low, as evidenced by increased evaluation counts and reduced optimization quality on tasks with dissimilar structures. In contrast, MFEA-SSG achieves superior performance with significantly lower computational costs. As shown in Table IV, the proposed method consistently outperforms baseline approaches in both solution quality and runtime. Particularly, the distilled student model achieves over $10\times$ speedup compared to the teacher model and other surrogate-based methods, while maintaining competitive performance. This demonstrates the effectiveness of combining generative modeling with knowledge distillation for cost-efficient multitask optimization. Beyond efficiency, MFEA-SSG shows strong adaptability across heterogeneous tasks. Its ability to consistently deliver high-quality solutions—regardless of task similarity or dimensional structure—reflects its generalization capability. This robustness can be attributed to the meta-learning-inspired training scheme, which encourages the model to learn transferable patterns rather than overfitting to task-specific features. The observed improvements in diverse task groups confirm the method's suitability for broad, resource-constrained optimization settings.

E. Ablation Study on Meta-Learning and Dimensional Independence in Generative Models

To quantitatively validate our claim that the meta-learning-inspired shuffling mechanism is crucial for performance, we conducted an ablation study with results presented in Table V. This experiment was designed to measure the impact of the artificial spatial bias introduced by naive, fixed-order reshaping. We compared the performance of our proposed shuffling

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2025.3617343

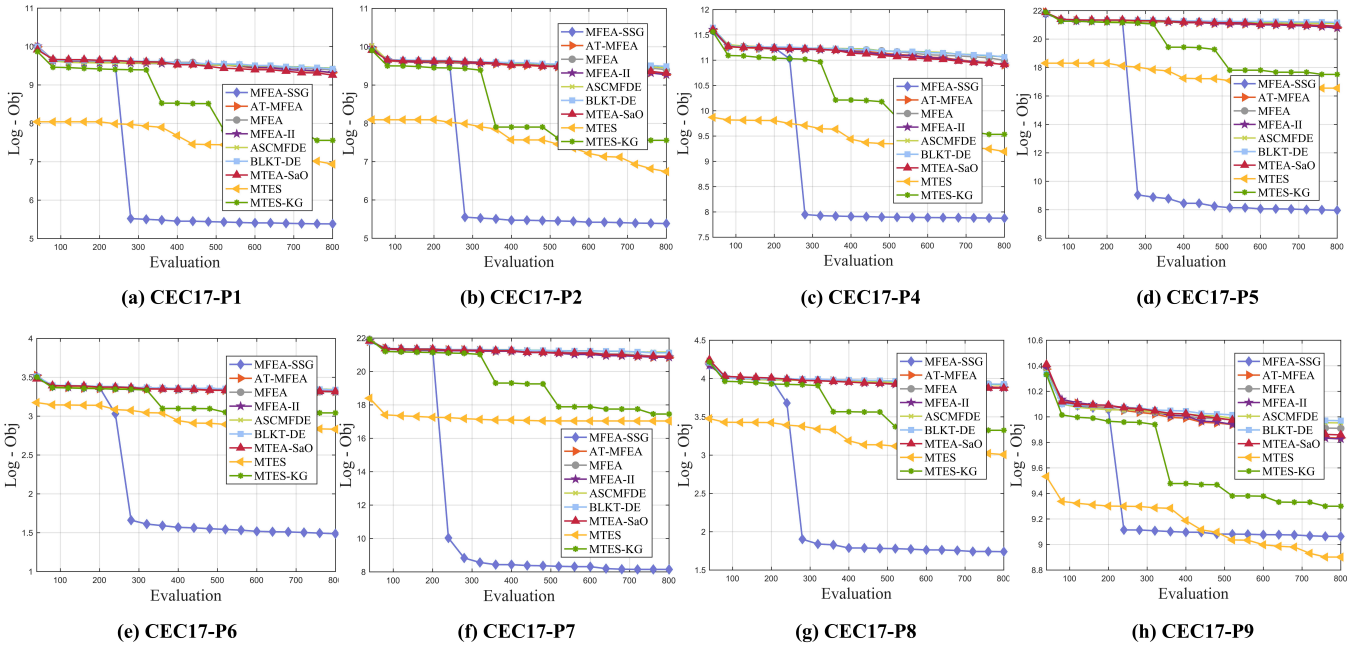


Fig. 2: The convergence trends of the average objective value of MFEA-SSG, AT-MFEA, MFEA, MFEA-II, ASCMFDE, BLKT-DE, MTEA-SaO, MTES and MTES-KG on the expensive MTSO for 20 independent runs.

approach (e.g., "Shuffle-5-10") against fixed-order baselines where the dimensions were not shuffled before reshaping (e.g., "5-10"). The results provide clear empirical evidence for our hypothesis. As shown in Table V, the shuffled configurations consistently and significantly outperform their fixed order counterparts. This strongly suggests that the original 50-dimensional individual representations do not possess a meaningful spatial structure, and that imposing a fixed arrangement inadvertently introduces a detrimental artificial structure that limits the model's ability to learn generalized features. Shuffling effectively eliminates this constraint, allowing the model to discover and utilize the underlying relationships between dimensions more freely, thus enhancing its generalization across tasks.

Next, we compared the distribution of generated individuals from models trained with two types of input: high-dimensional 1x50 data and data reshaped into 5x10 images after a random shuffle of dimensions. As shown in Fig. 3(a) and Fig. 3(c), the UMAP visualization of the self-attention layer output reveals that the 1x50 input model exhibits a more concentrated distribution with lower feature differentiation. It is important to clarify that the "50D→2D" label on the axes refers to the dimensionality reduction performed by UMAP for visualization purposes; the intrinsic dimensionality of the data remains 50, regardless of the input tensor's shape. In contrast, the shuffled 5x10 input model displays a richer, more dispersed distribution with several small clusters, indicating that the shuffled 5x10 input enables the model to capture finer-grained feature details and learn flexible relationships among dimensions, thus providing more robust feature support for generating individuals. Similarly, Fig. 3(b) and Fig. 3(d) illustrate the final output distributions, where the 5x10 model forms distinct clusters in UMAP space, showing enhanced

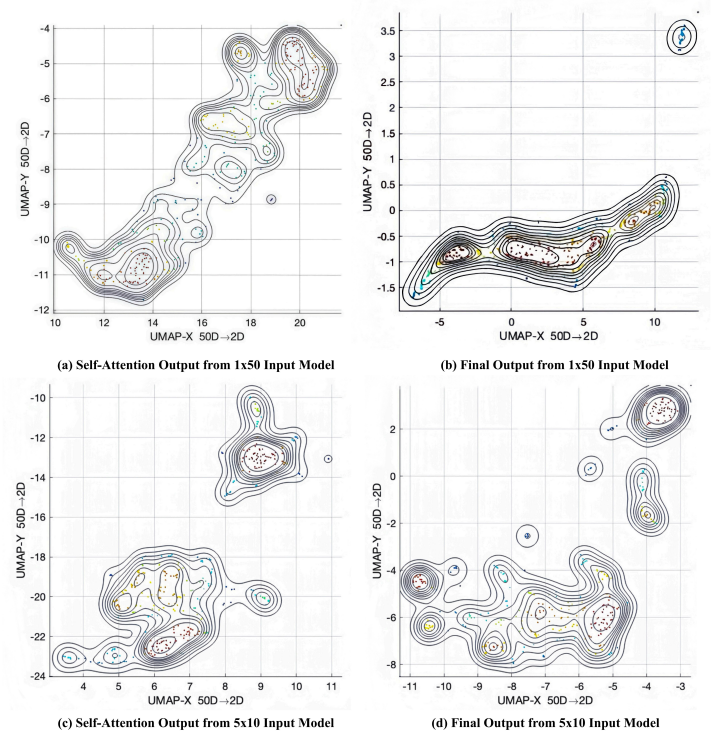


Fig. 3: Comparison of self-attention and final outputs between models trained with 1x50 and 5x10 inputs.

representation of data diversity and improved generalization across tasks. By comparison, the unshuffled 5x10 input could introduce redundant spatial relationships, leading the model to develop unnecessary dependencies; hence, it was excluded from this analysis.

TABLE V: PERFORMANCE COMPARISON OF GENERATIVE MODELS WITH SHUFFLED AND RESHAPED INPUT DATA ACROSS VARIOUS CONFIGURATIONS.

Category	Shuffle-5-10	Shuffle-10-5	Shuffle-2-25	Shuffle-25-2	5-10	10-5	2-25	25-2
P1-T1	1.0344e+00	**1.0287e+00** =	1.0322e+00 =	1.0343e+00 =	1.0815e+00 -	1.0786e+00 -	1.0798e+00 -	1.0726e+00 -
Std	1.90e-02	3.19e-02	1.18e-02	9.24e-03	1.13e-02	1.12e-02	8.36e-03	1.36e-02
P1-T2	4.8298e+02	4.8180e+02 =	4.7236e+02 =	**4.6757e+02** =	5.2310e+02 -	5.2955e+02 -	5.2418e+02 -	5.1712e+02 -
Std	2.30e+01	2.65e+01	3.67e+01	3.02e+01	3.12e+01	2.64e+01	2.25e+01	4.04e+01
P2-T1	**4.7781e+00**	4.9335e+00 =	4.7830e+00 =	4.7993e+00 =	6.3550e+00 -	6.1461e+00 -	6.1761e+00 -	6.0719e+00 -
Std	3.43e-01	3.50e-01	4.18e-01	2.87e-01	2.17e-01	4.12e-01	3.45e-01	3.36e-01
P2-T2	4.7869e+02	4.7856e+02 =	4.8444e+02 =	**4.7213e+02** =	5.2560e+02 -	5.2902e+02 -	5.1711e+02 -	5.1863e+02 -
Std	2.29e+01	2.47e+01	2.95e+01	2.54e+01	3.53e+01	2.87e+01	2.64e+01	2.61e+01
P3-T1	**2.1385e+01**	2.1393e+01 =	2.1399e+01 =	2.1395e+01 =	2.1385e+01 =	2.1403e+01 =	2.1391e+01 =	2.1397e+01 =
Std	4.89e-02	5.37e-02	3.98e-02	5.10e-02	7.16e-02	4.72e-02	4.58e-02	5.65e-02
P3-T2	1.7158e+04	1.7328e+04 =	**1.7094e+04** =	1.7345e+04 =	1.7138e+04 =	1.7401e+04 =	1.7395e+04 =	1.7294e+04 =
Std	7.99e+02	8.81e+02	7.26e+02	5.49e+02	4.84e+02	4.53e+02	6.61e+02	4.04e+02
P4-T1	**4.6809e+02**	4.7275e+02 =	4.7345e+02 =	4.7267e+02 =	5.3788e+02 -	5.3669e+02 -	5.2350e+02 -	5.1485e+02 -
Std	2.92e+01	3.18e+01	3.13e+01	2.29e+01	3.72e+01	1.98e+01	3.73e+01	2.82e+01
P4-T2	5.0624e+03	5.0266e+03 =	5.0794e+03 =	5.0504e+03 =	**4.3302e+03** +	4.8263e+03 +	4.8263e+03 +	4.5938e+03 +
Std	1.15e+02	1.32e+02	1.37e+02	1.36e+02	2.11e+02	2.26e+02	1.84e+02	1.76e+02
P5-T1	**5.1832e+00**	5.2991e+00 =	5.2778e+00 =	5.2832e+00 =	6.1924e+00 -	6.0162e+00 -	6.2279e+00 -	5.8985e+00 -
Std	2.47e-01	2.11e-01	2.93e-01	2.96e-01	2.65e-01	2.37e-01	3.00e-01	3.27e-01
P5-T2	**9.1078e+03**	1.3609e+04 -	1.0823e+04 =	1.1856e+04 -	4.5942e+04 -	4.4699e+04 -	4.2388e+04 -	3.6118e+04 -
Std	2.96e+03	7.32e+03	3.68e+03	3.46e+03	1.50e+04	1.45e+04	1.02e+04	7.55e+03
P6-T1	4.7978e+00	4.8185e+00 =	**4.6994e+00** =	4.8563e+00 =	6.1768e+00 -	6.1395e+00 -	6.2160e+00 -	5.9609e+00 -
Std	3.30e-01	2.88e-01	3.61e-01	3.14e-01	3.25e-01	2.55e-01	2.96e-01	3.49e-01
P6-T2	5.5305e+00	5.6646e+00 =	5.7154e+00 =	5.6137e+00 =	7.1348e+00 -	6.9867e+00 -	7.3518e+00 -	7.1941e+00 -
Std	5.15e-01	4.86e-01	6.37e-01	5.70e-01	4.80e-01	4.35e-01	5.12e-01	5.54e-01
P7-T1	1.2122e+04	**9.9868e+03** =	1.1335e+04 =	1.2132e+04 =	4.1763e+04 -	4.3344e+04 -	4.0890e+04 -	3.9056e+04 -
Std	4.70e+03	4.17e+03	4.09e+03	4.77e+03	1.39e+04	1.28e+04	1.63e+04	7.85e+03
P7-T2	4.8763e+02	**4.7278e+02** =	4.8535e+02 =	4.8643e+02 =	5.2486e+02 -	5.1512e+02 -	5.2926e+02 -	5.2334e+02 -
Std	2.58e+01	3.29e+01	2.50e+01	2.44e+01	3.65e+01	3.08e+01	3.23e+01	2.57e+01
P8-T1	1.0426e+00	1.0413e+00 =	1.0468e+00 =	**1.0392e+00** =	1.0926e+00 -	1.0897e+00 -	1.0902e+00 -	1.0815e+00 -
Std	1.13e-02	1.59e-02	2.00e-02	1.31e-02	1.99e-02	1.80e-02	2.05e-02	1.93e-02
P8-T2	1.2043e+01	1.1940e+01 =	1.1853e+01 =	**1.1837e+01** =	1.5165e+01 -	1.5281e+01 -	1.5035e+01 -	1.4545e+01 -
Std	8.41e-01	6.53e-01	8.14e-01	8.29e-01	8.70e-01	8.23e-01	9.89e-01	9.64e-01
P9-T1	4.8719e+02	4.7508e+02 =	4.7275e+02 =	**4.7158e+02** =	5.3422e+02 -	5.2600e+02 -	5.3362e+02 -	5.2599e+02 -
Std	3.60e+01	2.53e+01	3.04e+01	3.44e+01	3.07e+01	3.12e+01	2.84e+01	3.36e+01
P9-T2	1.7349e+04	**1.6864e+04** =	1.7288e+04 =	1.7303e+04 =	1.7268e+04 =	1.7135e+04 =	1.7377e+04 =	1.7201e+04 =
Std	5.66e+02	9.23e+02	7.79e+02	8.21e+02	6.33e+02	6.13e+02	5.50e+02	6.81e+02
+ / - / =	Base	0 / 1 / 17	0 / 0 / 18	0 / 1 / 17	1 / 14 / 3	1 / 14 / 3	1 / 14 / 3	1 / 14 / 3

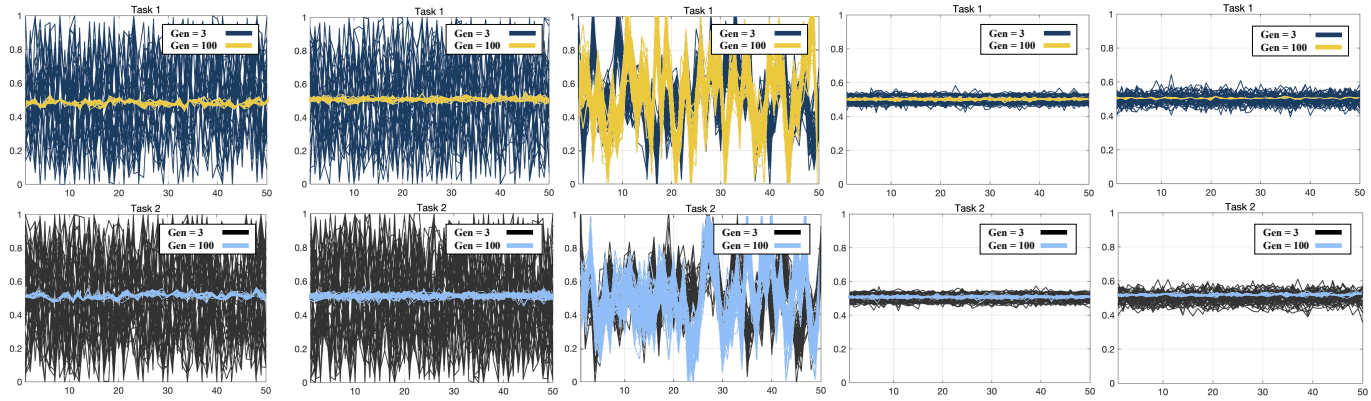
By disrupting the task-specific dimension order through shuffling, the model's training is no longer bound by misleading patterns imposed by an arbitrary variable ordering. This aligns with the meta-learning principle, as the model learns to generalize from varied and recomposed samples, making it adaptable to new tasks. This highlights how the artificial dependencies imposed by a fixed dimension order can mislead the model into task-specific local optima, reducing its ability to generalize. The shuffling encourages the model to learn more diverse and robust feature representations, which improves its ability to explore the decision space and reduces the risk of overfitting to task-specific patterns.

F. Comparison of Teacher and Single-Step Student Models Before and After Knowledge Distillation

To evaluate the impact of knowledge distillation, we conduct ablation experiments comparing the teacher model (MFEA-SSG-Teacher) and the distilled student model (MFEA-SSG-Student) under same evolutionary conditions. Specifically, we compare their performance against three baseline algorithms: MFEA, an evolutionary multitasking baseline; G-MFEA, a classic expensive multitasking optimization baseline; and MTES, an evolutionary strategy-based multitasking baseline. The effectiveness of knowledge distillation is assessed by analyzing decision variable distributions at both early and later evolutionary stages, as well as by comparing computational efficiency. As shown in Fig. 4, we visualize the decision variable distributions at Gen = 3 (very early stage) and Gen = 100 (later stage). The results indicate that for MFEA, G-MFEA, and MTES, the distributions at Gen = 3 remain nearly uniform, similar to the initial population, suggesting minimal progress in the early stage. These algorithms only begin to show convergence trends as evolution progresses, reaching more structured distributions at Gen = 100. In contrast, both MFEA-SSG-Teacher and MFEA-SSG-Student exhibit clear convergence patterns even at Gen = 3, indicating a significantly faster adaptation process. Notably, the student model achieves a similar distribution to the teacher model despite operating with significantly lower computational overhead.

The efficiency advantage of knowledge distillation is further confirmed by the runtime comparison in Fig. 5, where execution time is plotted in logarithmic scale due to the huge difference between models. As expected, MFEA-SSG-Student demonstrates at least an order-of-magnitude reduction in computational cost compared to the teacher model, aligning with the complexity analysis presented in previous sections. Despite this drastic reduction in computational overhead, Fig. 4 shows that the student model maintains comparable optimization

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2025.3617343



(a) MFEA (b) G-MFEA (c) MTES (d) MFEA-SSG- Teacher (e) MFEA-SSG-Student

Fig. 4: Comparison of decision variables in advanced multitask algorithms under different evolutionary generations.

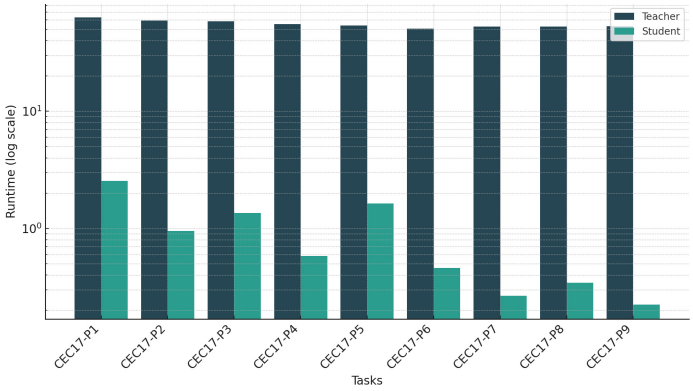


Fig. 5: Runtime comparison of teacher and student model.

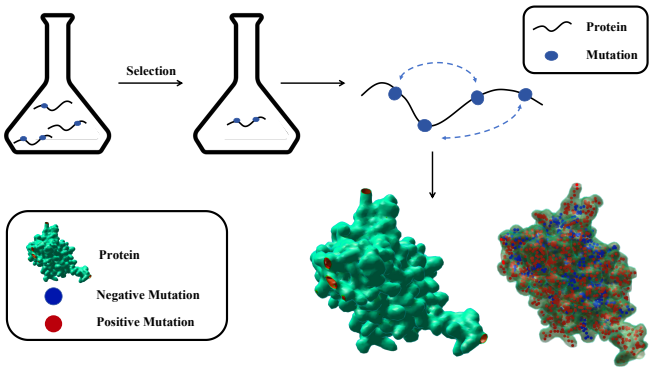


Fig. 6: Directed evolution of protein.

performance, particularly in early-stage evolution where it quickly learns meaningful decision distributions. These results highlight the benefits of knowledge distillation in MFEA-SSG-Student. By significantly reducing computational cost while preserving strong convergence behavior, the student model enables efficient optimization in expensive multitask scenarios. The ability to achieve rapid adaptation with limited evolutionary iterations makes it particularly valuable when computational resources are constrained.

G. A Case Study: Optimizing Protein Mutation Prediction with MFEA-SSG

Protein mutation prediction is a key task in protein engineering and bioinformatics [45]–[47], aiming to identify mutations that enhance protein function or stability while filtering out those that lead to functional loss. Fig. 6 provides a visual representation to better illustrate the problem under investigation. Since the effects of mutations involve complex relationships across sequence, structure, and function [48], traditional experimental validation methods, such as wet-lab experiments, are not only time-consuming and labor-intensive but also costly. This challenge closely resembles the expensive evaluation process in ExMTOPs. Therefore, efficiently predicting mutation impacts with limited experimental resources has become a critical challenge in this field.

In this study, inspired by meta-learning, we model the protein mutation prediction problem as a multitask optimization problem under the MFEA-SSG framework. Specifically, we decompose mutation prediction into two sub-tasks: positive mutation learning, which identifies beneficial mutations for protein function, and negative mutation learning, which detects mutations leading to functional loss. We represent protein sequences using one-hot encoding and incorporate protein evolutionary information—the frequency of amino acid occurrences in multiple sequence alignment [52] (MSA)—as additional features. This evolutionary information functions similarly to probability modeling of individual decision variables, enabling the model to extract more generalizable patterns from existing mutation data and avoid overfitting to specific mutation cases. Within the MFEA-SSG framework, we leverage cross-task information sharing to enable simultaneous learning of both positive and negative mutation patterns while enhancing the model's ability to capture mutation effects through shared features. By adopting this approach, we aim to train a model that not only learns evolutionary patterns of mutations effectively but also improves biological plausibility and generalization in mutation generation, thereby reducing the need for experimental evaluations and providing an efficient computational tool for protein design and optimization.

For our experiments, we utilize the green fluorescent protein

(GFP) mutation dataset provided by Karen S. Sarkisyan et al. [49], which contains sequences of numerous GFP mutants along with their corresponding fluorescence intensity measurements. Protein sequences are represented as one-hot encoded vectors of length 238, where each amino acid is encoded as a 20-dimensional vector, resulting in a data format of (N, 238, 20), where N is the number of samples. Additionally, amino acid occurrence frequencies are extracted from MSA data and normalized to the range [0,1] to assist in learning mutation patterns. The dataset is split into 80% training, 10% validation, and 10% testing subsets. Positive and negative samples are processed within the MFEA-SSG framework, where training is conducted using a task-sharing mechanism. Throughout training, the model learns both positive and negative mutation patterns in a balanced manner while leveraging evolutionary information to construct probability distributions, guiding the model to focus on amino acid positions with a higher likelihood of mutation and reducing random mutations at conserved sites. A diffusion-based generative model is employed to simulate the mutation generation process, progressively reconstructing potential mutation structures based on the learned patterns to generate biologically plausible protein sequences.

TABLE VI: COMPARISON OF LEVENSHTEIN DISTANCE AND JACCARD SIMILARITY ACROSS METHODS.

Algorithm	Levenshtein Distance	Jaccard Similarity
EMT-GS	4.2	0.85
MFEA-VC	6.1	0.78
MFEA-SSG	**7.5**	**0.72**

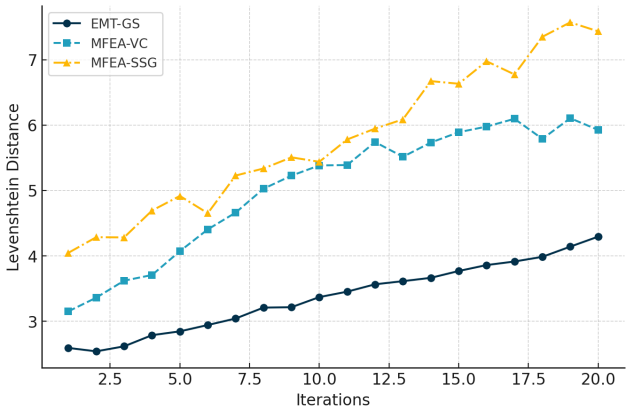


Fig. 7: Levenshtein distance evolution over iterations.

To assess the quality of generated protein mutations, we evaluated MFEA-SSG and two other generative multitask algorithms based on three key criteria: biological rationality, utilization of evolutionary information, and pattern stability. We specifically analyzed their performance on single- and double-point mutation tasks. To quantify performance, we employed Levenshtein Distance to measure sequence-level variation and Jaccard Similarity to evaluate the novelty of generated mutations. The results (Table VI, Figs. 7-8) reveal distinct behaviors tied to each algorithm's generative mechanism. EMT-GS primarily replicates training data, resulting in high similarity but low diversity and a risk of pattern collapse.

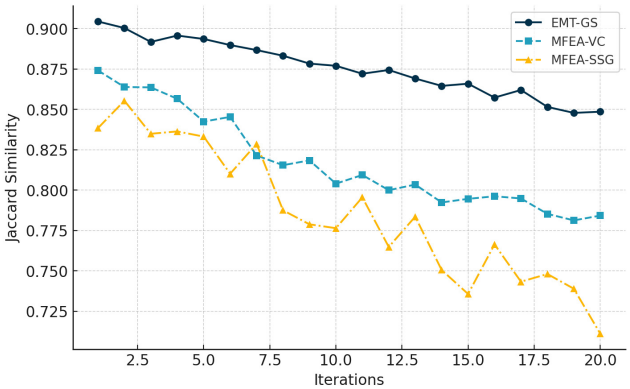


Fig. 8: Jaccard similarity evolution over iterations.

MFEA-VC offers an improvement by modeling mutation distributions with latent variables, yet produces conservative patterns with limited innovation. In contrast, MFEA-SSG's diffusion-based model enables sequences to evolve into biologically meaningful patterns. Its inter-task information sharing enhances biological plausibility and the use of evolutionary data, effectively preventing pattern collapse while generating a diverse and stable set of mutations.

TABLE VII: COMPARISON OF SINGLE MUTATION OVERLAP, POSITIVE RATE, AND WET LAB SCORES ACROSS DIFFERENT ALGORITHMS.

Algorithm	Mutation Overlap	Positive Rate	Wet Lab Score
EMT-GS	**113**	60.8%	2.46
MFEA-VC	94	63.5%	2.52
MFEA-SSG	81	**71.4%**	**2.59**

TABLE VIII: COMPARISON OF DOUBLE MUTATION OVERLAP, POSITIVE RATE, AND WET LAB SCORES ACROSS DIFFERENT ALGORITHMS.

Algorithm	Mutation Overlap	Positive Rate	Wet Lab Score
EMT-GS	**87**	61.9%	2.48
MFEA-VC	72	64.8%	2.56
MFEA-SSG	63	**69.1%**	**2.63**

Since protein engineering often targets a few amino acid sites, we further analyzed the models' ability to generate effective single- and double-point mutations. We measured the overlap with the test set, positive mutation rate, and average wet lab score for 5000 generated sequences per algorithm (Tables VII-VIII). EMT-GS achieved the highest overlap, confirming its strong memorization of existing patterns but at the cost of the lowest positive rate and wet lab score. MFEA-VC demonstrated a balanced trade-off, capturing general trends for moderate performance across all metrics. Conversely, MFEA-SSG exhibited the lowest overlap, reflecting high novelty. Crucially, this diversity translated into the highest positive mutation rate and wet lab scores, demonstrating its superior capability in generating functionally promising and biologically meaningful mutations.

V. Conclusion

This paper presents MFEA-SSG, a meta-learning-inspired single-step generative model tailored for ExMTOPs. By learning the distribution of high-quality solutions through a diffusion-based generative model, MFEA-SSG significantly reduces reliance on frequent function evaluations. The model adopts a variable-shuffling and reshaping strategy to disrupt task-specific dependencies, enabling the learning of transferable structures across tasks. To further lower inference cost, knowledge distillation compresses the complex diffusion model into a lightweight student model capable of single-step generation. Comprehensive experiments across general multitask benchmarks and a real-world protein mutation prediction task confirm that MFEA-SSG consistently achieves faster convergence and higher-quality solutions under tight evaluation budgets. Its generative nature, coupled with strong generalization ability, allows it to effectively handle heterogeneous and high-dimensional optimization problems.

Future work will explore enhancing the transparency and trustworthiness of the generative model to support more robust deployment in complex, evaluation-constrained scenarios. Drawing inspiration from principles in Explainable AI, we aim to investigate how to better align the model's behavior with task-level understanding, thereby improving its reliability in real-world optimization problems.

References

[1] J. Ding, C. Yang, Y. Jin, et al, "Generalized multitasking for evolutionary optimization of expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 44-58, 2017.

[2] F. Ming, W. Gong, L. Wang, et al, "Constrained multiobjective optimization via multitasking and knowledge transfer," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 77-89, 2022.

[3] F. Ming, W. Gong, L. Gao, "Adaptive auxiliary task selection for multitasking-assisted constrained multi-objective optimization," *IEEE Computational Intelligence Magazine*, vol. 18, no. 2, pp. 18-30, 2023.

[4] Z. Li, X. Ma, Z. Zhu, et al, "Expensive Optimization Based on Evolutionary Multi-Tasking and Hybrid Restart Strategy," *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1-9, 2024.

[5] Y. Li, W. Gong, "Multiobjective multitask optimization with multiple knowledge types and transfer adaptation," *IEEE Transactions on Evolutionary Computation*, vol. 29, no. 1, pp. 205-216, 2025.

[6] S. Tan, Y. Wang, G. Sun, et al, "A Surrogate-Assisted Evolutionary Framework for Expensive Multitask Optimization Problems," *IEEE Transactions on Evolutionary Computation*, 2024.

[7] J. Liu, Y. Wang, G. Sun, and T. Pang, "Multisurrogate-assisted ant colony optimization for expensive optimization problems with continuous and categorical variables," *IEEE Transactions on Cybernetics,* vol. 52, no. 11, pp. 11348-11361, 2021.

[8] L. Yu, Z. Meng, H. Zhu, "A Hierarchical Surrogate-Assisted Differential Evolution With Core Space Localization," *IEEE Transactions on Cybernetics,* 2024.

[9] J. Liu, Y. S. Ong, and M. Wong, "Finding Sets of Pareto Sets in Real-World Scenarios–A Multitask Multiobjective Perspective," *IEEE Congress on Evolutionary Computation (CEC)* , pp. 1-8, June. 2024.

[10] S. Tan, Y. Wang, G. Sun and T. Pang, "A Surrogate-Assisted High-Dimensional Mixed-Variable Evolutionary Framework and Its Application to Vehicle Lightweighting Design," *IEEE Transactions on Evolutionary Computation,* 2025.

[11] J. Liu, Y. Wang, G. Sun, and T. Pang, "Constrained Evolutionary Bayesian Optimization for Expensive Constrained Optimization Problems With Inequality Constraints," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024.

[12] J. Shen, H. Dong, P. Wang, et al, "Surrogate-Assisted Adaptive Knowledge Transfer for Expensive Multitasking Optimization," *2024 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1-8, 2024.

[13] H. Wang, L. Feng, Y. Jin, et al, "Surrogate-assisted evolutionary multitasking for expensive minimax optimization in multiple scenarios," *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 34-48, 2021.

[14] J. Liu, A. Gupta, C. Ooi and Y. S. Ong, "ExTrEMO: Transfer evolutionary multiobjective optimization with proof of faster convergence," *IEEE Transactions on Evolutionary Computation,* 2024.

[15] J. Liu, A. Gupta and Y. S. Ong, "Bayesian inverse transfer in evolutionary multiobjective optimization," *ACM Transactions on Evolutionary Learning,* vol. 4, no. 4, pp. 1-27, 2025.

[16] J. Liu, A. Gupta, Y. S. Ong and P. S. Tan, " Inverse multiobjective optimization by generative model prompting," *2024 IEEE Conference on Artificial Intelligence (CAI),* pp. 737-740, June. 2024.

[17] S. Liu, J. Yan, H. Wang, et al, "Morphology Evolution for Embodied Robot Design With a Classifier-Guided Diffusion Model," *IEEE Transactions on Evolutionary Computation*, pp. 1-1, 2025.

[18] Y. Huang, L. Feng, A. Qin, et al, "Toward large-scale evolutionary multitasking: A GPU-based paradigm," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 3, pp. 585-598, 2021.

[19] S. Huang, J. Zhong, W. Yu, "Surrogate-assisted evolutionary framework with adaptive knowledge transfer for multi-task optimization," *IEEE transactions on emerging topics in computing*, vol. 9, no. 4, pp. 1930-1944, 2019.

[20] L. Yu, Z. Meng, L. Kong, et al., "Surrogate-assisted differential evolution: A survey," *Swarm and Evolutionary Computation,* 2025, 94: 101879.

[21] X. Wu, S. Liu, Q. Lin, et al. "Evolutionary Multitasking With Adaptive Knowledge Transfer for Expensive Multiobjective Optimization," *IEEE Transactions on Evolutionary Computation*, 2024.

[22] 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, Jun. 2016.

[23] R. Salakhutdinov, "Learning deep generative models," *Annual Review of Statistics and Its Application,* vol. 2, no. 1, pp. 361-385, 2015.

[24] G. M. Harshvardhan, M. K. Gourisaria, M. Pandey, et al., "A comprehensive survey and analysis of generative models in machine learning," *Computer Science Review,* vol. 38, pp. 100285, 2020.

[25] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, et al., "An introduction to variational methods for graphical models," *Machine learning,* vol. 37, pp. 183-233, 1999.

[26] Y. Deldjoo,Z. He, J. McAuley, et al., "A review of modern recommender systems using generative models (gen-recsys)," *the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining,* pp. 6448-6458, 2024.

[27] R. Wang, X. Feng and H. Yu, "Contrastive variational auto-encoder driven convergence guidance in evolutionary multitasking," *Applied Soft Computing,* vol. 163, pp. 111883, 2024.

[28] Z. Liang, Y. Zhu, X. Wang, et al., "Evolutionary multitasking for optimization based on generative strategies," *IEEE Transactions on Evolutionary Computation,* vol. 27, no. 4, pp. 1042-1056, 2022.

[29] R. Wang, X. Feng, H. Yu, EMK. Lai, "Residual Learning Inspired Crossover Operator and Strategy Enhancements for Evolutionary Multitasking," *arXiv preprint arXiv:2503.21347*, 2025.

[30] G. Hinton, O. Vinyals, J. Dean, "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531, 2015.

[31] G. Wang, P. Zhao, Y. Shi, et al., "Generative model-based feature knowledge distillation for action recognition," *Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 38, no. 14, pp. 15474-15482, 2024.

[32] A. Aguinaldo, P. Y. Chiang, A. Gain, et al., "Compressing gans using knowledge distillation," arXiv preprint arXiv:1902.00159, 2019.

[33] T. Yin, M. Gharbi, R. Zhang, et al., "One-step diffusion with distribution matching distillation," *IEEE/CVF conference on computer vision and pattern recognition,* pp. 6613-6623, 2024.

[34] J. J. Dongarra, "Performance of various computers using standard linear equations software," ACM SIGARCH Computer Architecture News, vol. 20, no. 3, pp. 22-44, 1992.

[35] Y. Li, W. Gong, F. Ming, et al., " MToP: A MATLAB Optimization Platform for Evolutionary Multitasking". *arXiv:2312.08134*, 2023.

[36] X. Xue et al., "Affine Transformation-Enhanced Multifactorial Optimization for Heterogeneous Problems," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 6217-6231, Jul. 2022.

[37] 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*, vol. 24, no. 1, pp. 69-83, Feb. 2020.

[38] Z. Tang, M. Gong, Y. Wu, et al., "Regularized evolutionary multitask optimization: Learning to intertask transfer in aligned subspace," *IEEE*

Transactions on Evolutionary Computation, vol. 25, no. 2, pp. 262-276, 2020.

[39] Y. Jiang, Z. Zhan, K. Tan, et al, "Block-level knowledge transfer for evolutionary multitask optimization," *IEEE Transactions on Cybernetics*, vol. 54, no. 1,pp: 558-571, 2023.

[40] Y. Li, W. Gong, and S. Li, "Multitasking optimization via an adaptive solver multitasking evolutionary framework," *Information Sciences*, vol. 630, pp. 688–712, 2023.

[41] L. Bai, W. Lin, A. Gupta, et al., "From multitask gradient descent to gradient-free evolutionary multitasking: A proof of faster convergence," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 8561-8573, 2021.

[42] Y. Li, W. Gong, S. Li, "Multitask evolution strategy with knowledge-guided external sampling," *IEEE Transactions on Evolutionary Computation,* 2023.

[43] M. Pelikan, "Bayesian optimization algorithm," *Hierarchical Bayesian optimization algorithm: toward a new generation of evolutionary algorithms,* pp. 31-48, 2005.

[44] J. Liu, Y. Wang, G. Sun, et al., "Solving highly expensive optimization problems via evolutionary expected improvement," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 53, no. 8, pp. 4843-4855, 2023.

[45] Y. Tan, Z. Zhang, M. Li, et al., "MedChatZH: A tuning LLM for traditional Chinese medicine consultations," *Computers in biology and medicine,* vol. 172, pp. 108290, 2024.

[46] Y. Tan, C. Liu, J. Gao, et al., "VenusFactory: A Unified Platform for Protein Engineering Data Retrieval and Language Model Fine-Tuning," *arXiv preprint arXiv:2503.15438*, 2025.

[47] M. Li, Y. Tan, X. Ma, et al., "ProSST: Protein Language Modeling with Quantized Structure and Disentangled Attention," *Advances in Neural Information Processing Systems*, vol. 39, pp. 35700-35726, 2024.

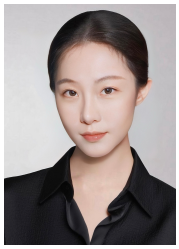
[48] Y. Tan, R. Wang, B. Wu, et al., "Retrieval-enhanced mutation mastery: Augmenting zero-shot prediction of protein language model," *arXiv preprint arXiv:2410.21127*, 2024.

[49] K. S. Sarkisyan, A. S. Goryashchenko, P. V. Lidsky, et al., "Green fluorescent protein with anionic tryptophan-based chromophore and long fluorescence lifetime," *Biophysical journal,* vol. 109, no. 2, pp. 380-389, 2015.

[50] B. Berger, M. Waterman, Y. Yu, "Levenshtein distance, sequence comparison and biological database search," *IEEE transactions on information theory,* vol. 67, no. 6, pp. 3287-3294, 2020.

[51] N. Chung, B. Miasojedow, M. Startek, et al., "Jaccard/Tanimoto similarity test and estimation methods for biological presence-absence data," *BMC bioinformatics,* vol. 20, no. 15, pp. 644, 2019.

[52] Y. Zhang, M. Lang, J. Jiang, et al. Multiple sequence alignment-based RNA language model and its application to structural inference[J]. Nucleic Acids Research, 2024, 52(1): e3-e3.



Ruilin Wang is currently a Ph.D. candidate with the School of Information Science and Engineering, East China University of Science and Technology, Shanghai, China. She is also a joint Ph.D. student with the Department of Data Science and Artificial Intelligence, Auckland University of Technology, Auckland, New Zealand. Her research interests include evolutionary multitasking and its integration with deep learning.

Xiang Feng received the M.Sc. degree in System Engineering from Naval Engineering University, China, in 2003. She received the Ph.D. degree in Control Theory and Engineering from East China University of Science and Technology, China, in 2006. She worked as a postdoctoral fellow in the Department of Computer Science of the University of Hong Kong from 2006 to 2008. She is presently a professor of the Department of Computer Science and Engineering, East China University of Science and Technology. Her research interests include distributed swarm intelligence and evolutionary computing, integration learning and integration optimization, big data intelligence.

Huiqun Yu received the M.Sc. and Ph.D. degrees in Computer Science from the East China University of Science and Technology and Shanghai Jiaotong University, in 1992 and 1995, respectively. He is presently a professor of the Department of Computer Science and Engineering, East China University of Science and Technology. His main research interests include software engineering, trustworthy computing and big data intelligence.

Yang Tan received the M.Sc degree from the East China University of Science and Technology in July 2025, and B.S. degree from the East China University of Science and Technology in July 2022. He is currently a Ph.D. student in the school of computer science at Shanghai Jiao Tong University. His research interests include bioinformatics, protein language model, and graph neural network.

Edmund Lai received the Bachelor of Engineering with Honours and the Ph.D. degrees, both in Electrical Engineering, from the University of Western Australia, in 1982 and 1991 respectively. He is a professor in the Department of Data Science and Artificial Intelligence, Auckland University of Technology (AUT), New Zealand. He is also the Director of AUT AI Research Centre. His research interests include scheduling and optimization, intelligent control, and multi-agent systems.