A Multi-Objective Optimization Framework for Adaptive Weighting in Physics-Informed Machine Learning

Guoquan Wu, Zhe Wu

Department of Chemical and Biomolecular Engineering, National University of Singapore, 117585, Singapore
wu.guoquan@u.nus.edu, wuzhe@nus.edu.sg

Abstract

Training physics-informed neural networks (PINNs) can be viewed as a multi-task optimization problem, where data-driven and physics-driven loss functions must be simultaneously minimized, despite the potential competition between them. Manually tuning the weight coefficients for various loss terms in PINNs is often time-consuming and lacks a systematic approach. To address this challenge, this work proposes an adaptive loss balancing framework for PINNs, using multi-objective optimization (MOO) algorithms to dynamically balance competing loss terms during training. Specifically, the Non-dominated Sorting Genetic Algorithm II (NSGA-II) is integrated into the PINN training process to explore the Pareto front of the multiple objectives. A novel variance-aware relative improvement (VARI) weighting method is proposed to translate Pareto-optimal information into adaptive loss weights. The proposed MOO-VARI method is validated through several examples, where the results show that the MOO-VARI PINN consistently outperforms standard PINN and other state-of-the-art adaptive weighting strategies in terms of convergence speed, predictive accuracy, and parameter estimation performance.

Introduction

Physics-informed neural networks (PINNs) have emerged as a rapidly advancing paradigm within the intersection of machine learning and engineering systems (Karniadakis et al. 2021). By incorporating physical laws into the loss function of neural networks, PINNs can guide model training toward physically consistent solutions rather than relying purely on data-driven approaches. In contrast to traditional machine learning methods, which might require a large amount of labeled data or risk violating physical constraints, PINNs can better handle scenarios involving limited data, complex boundary conditions, or noisy measurements.

PINNs typically incorporate multiple loss terms to constrain the solution space, including residual loss from partial differential equation (PDE), initial condition loss, and boundary condition loss (Raissi, Perdikaris, and Karniadakis 2019; Hao et al. 2024). The different loss terms collectively guide the network to approximate the true solution while satisfying physical constraints (Müller and Zeinhofer 2023). To balance the individual contributions of each term to the overall gradient, one usually introduces scaling factors. Previous studies have indicated that the training efficiency of PINNs is highly dependent on the weights associated with the different loss terms (Wang, Teng, and Perdikaris 2021; Hwang and Lim 2024; Rathore et al. 2024). Manual adjustment of these weights through grid or random search can be time consuming, especially when the number of loss terms increases (Krishnapriyan et al. 2021). Although several studies have explored methods to adjust the relative importance of different loss terms (Elhamod et al. 2022; Van Der Meer, Oosterlee, and Borovykh 2022), these approaches often remain non-adaptive across different training stages and can therefore miss the ideal balance as the network develops.

Recent studies have focused on adaptive methods for weight selection (Wang, Yu, and Perdikaris 2022; Liu et al. 2024; McClenny and Braga-Neto 2023; Xiang et al. 2022). However, these existing adaptive strategies remain fundamentally scalarisation-based, limiting their ability to explicitly consider trade-offs among conflicting objectives at different training stages. The difficulty is further exacerbated under parameter uncertainty, such as in engineering applications, where the first-principles model parameters might be partially unknown. In these cases, the weights chosen by these adaptive methods can overfit incomplete physical constraints, leading to suboptimal solutions that do not satisfy observed data or accurate governing equations. This highlights the necessity for a more flexible and adaptive framework that adjusts loss weights while respecting inter-objective trade-offs and parametric uncertainties.

From an optimization perspective, the training of a PINN can be formulated as a multi-objective optimization (MOO) problem, where each loss term represents a unique objective to be minimized. Therefore, MOO methods may provide a more comprehensive exploration of the solution space and provide valuable Pareto frontier information, which highlights the trade-offs between competing objectives (Wang et al. 2020). Despite this potential, adaptively adjusting the weights of each loss term through multi-objective optimization techniques has not been investigated at this stage. This gap presents a significant opportunity to improve the efficiency of PINN training and the predictive performance through a systematic method of loss balancing.

Motivated by the above considerations, in this work,

we develop an adaptive weighting framework for physics-informed machine learning based on MOO to tune the loss term weights. Specifically, we treat different loss terms as separate objectives in a multi-objective setting. The non-dominated sorting genetic algorithm II (NSGA-II) is employed to jointly optimize these objectives, discovering a Pareto frontier of solutions that reflect different trade-offs among the multiple loss terms. Based on Pareto-optimal solutions, we explore the competition and synergy between different objectives and propose a variance-aware relative improvement (VARI) method that dynamically assigns weights to each loss term during training. Through several case studies including both PDEs and ordinary differential equations (ODEs), we demonstrate the effectiveness of the proposed framework in improving the performance of PINNs.

Related Work

Physics-Informed Neural Networks. PINN has attracted significant interest in a variety of fields, such as fluid dynamics (Cai et al. 2021a), material science (Wang et al. 2024), and engineering domains (Zobeiry and Humfeld 2021; Cai et al. 2021b). Despite the demonstrated success of PINNs, it still faces challenges when solving highly nonlinear PDEs, which may show slower convergence rates and get stuck at the local optimal solution (Krishnapriyan et al. 2021). To accelerate the learning of PINNs, extensive research has focused on sampling schemes (Matsubara and Yaguchi 2025; Yang, Qiu, and Fu 2023) and allocation of loss weights (Liu et al. 2024). For example, Relative Loss Balancing with Random Lookback (ReLoBRaLo) is developed to periodically adjust loss weights based on historical trends and relative magnitudes of losses (Bischof and Kraus 2025).

Weighting method for MOO. MOO is a powerful tool for handling conflicting objectives in engineering design and decision-making problems (Gunantara 2018; Sharma and Kumar 2022). A variety of algorithmic families (Mirjalili, Jangir, and Saremi 2017; Ye et al. 2022) are used to approximate Pareto fronts in non-convex landscapes and have seen extensive use in engineering design. Typically, solutions on the Pareto front represent trade-offs among competing objectives, without explicit prioritization. Therefore, decision-makers must introduce objective weighting methods to select a final solution that best aligns with practical or domain-specific preferences (Wang et al. 2020; Ye et al. 2025). Among existing approaches, criteria importance through intercriteria correlation (CRITIC) method (Diakoulaki, Mavrotas, and Papayannakis 1995), statistical variance methods (Yusop et al. 2015), and entropy method (Li, Liu, and Li 2014) are frequently used to determine relative importance for individual objectives.

Preliminaries

Physics-Informed Neural Networks

Consider a nonlinear PDE of the general form:

$$\begin{aligned} &u_t + \mathcal{N}_x[u;k] = 0, \quad x \in \Omega, t \in [0,T], \\ &u(x,0) = I(x), \qquad x \in \Omega, \\ &u(x,t) = B(x,t), \quad x \in \partial\Omega, t \in [0,T], \end{aligned} \tag{1}$$

where $x \in \Omega$ and $t \in [0,T]$ are the spatial and time coordinates, respectively. $\mathcal{N}_x$ is a nonlinear differential operator. $u(x,t)$ is the continuous latent function to be found under the specific initial condition $I(x)$ and the boundary condition $B(x,t)$. k is a set of unknown parameters in the nonlinear differential operator $\mathcal{N}_x$.

In PINNs, the latent solution $u(x,t)$ is approximated by a neural network $\hat{u}(x,t;\theta,k)$ with trainable parameters θ and k. Specifically, the forms of the loss terms are defined as follows:

$$\begin{aligned} &\mathcal{L}_r(\theta,k) = \tfrac{1}{N_r}\textstyle\sum_{j=1}^{N_r} \left\| \frac{\partial \hat{u}}{\partial t}(x_r^j, t_r^j; \theta, k) + \mathcal{N}_x\left[\hat{u}(x_r^j, t_r^j; \theta, k)\right] \right\|^2, \\ &\mathcal{L}_{BC}(\theta,k) = \tfrac{1}{N_b}\textstyle\sum_{j=1}^{N_b} \left\| \hat{u}\left(x_b^j, t_b^j; \theta, k\right) - B(x_b^j, t_b^j) \right\|_2^2, \\ &\mathcal{L}_{IC}(\theta,k) = \tfrac{1}{N_i}\textstyle\sum_{j=1}^{N_i} \left\| \hat{u}\left(x_i^j, 0; \theta, k\right) - I(x_i^j) \right\|_2^2, \\ &\mathcal{L}_{data}(\theta,k) = \tfrac{1}{N_{data}}\textstyle\sum_{i=1}^{N_{data}} \left\| \hat{u}\left(x_{data}^j, t_{data}^j; \theta, k\right) - u_{data}^j \right\|_2^2, \end{aligned} \tag{2}$$

where $\mathcal{L}_r$ is the residual loss term, which measures how accurately the PINN output $\hat{u}(x,t;\theta,k)$ satisfies the governing equation at a set of interior collocation points $\{(x_r^j, t_r^j)\}_{j=1}^{N_r}$. Minimizing this residual ensures that the neural network remains closely aligned with the underlying physics captured by the governing equations. The boundary condition loss $\mathcal{L}_{BC}$ uses the boundary data $\{(x_b^j, t_b^j), B(x_b^j, t_b^j)\}_{j=1}^{N_b}$ to measure the discrepancy between the PINN output $\hat{u}(x,t;\theta,k)$ and the prescribed boundary condition $B(x,t)$, which ensures that the network predictions consistently match boundary constraints. Similarly, the initial condition loss $\mathcal{L}_{IC}$ renders the predictions to initial data $\{x_i^j, I(x_i^j)\}_{j=1}^{N_i}$. Additionally, the data loss term $\mathcal{L}_{data}$ can be included to evaluate the discrepancy between neural network predictions and measured values at sample data $\{x_{data}^j, t_{data}^j, u_{data}^j\}_{j=1}^{N_{data}}$ if additional measurements are available from experiments, sensor readings, or other solutions. The overall loss function for training the PINN often takes the form:

$$\begin{aligned} \mathcal{L}(\theta,k) = &\lambda_r \mathcal{L}_r(\theta,k) + \lambda_b \mathcal{L}_{BC}(\theta,k) \\ &+ \lambda_i \mathcal{L}_{IC}(\theta,k) + \lambda_d \mathcal{L}_{data}(\theta,k) \end{aligned} \tag{3}$$

where λ_r, λ_b, λ_i, and λ_d are loss weights that control the relative importance of each loss term during the optimization process. It should be noted that PINNs have not only demonstrated success in solving PDEs, but have also been widely applied to ODEs, which cover a broad range of problems in industrial domains. These applications include, but are not limited to, solving chemical kinetics (Ji et al. 2021), studying the dynamics of the system (Sun et al. 2025), and modeling the power system (Misyris, Venzke, and Chatzivasileiadis 2020).

Methodology

Given that the objectives in Eq. 3 are trained simultaneously, the training process can be viewed as a multi-objective optimization problem, where the goal is to find a set of solutions that optimize multiple (conflicting) objectives simultaneously. Therefore, in this section, we develop an adaptive loss balancing framework from the perspective of multi-objective optimization to determine the loss weights during the training process, which is illustrated in Fig. 1.

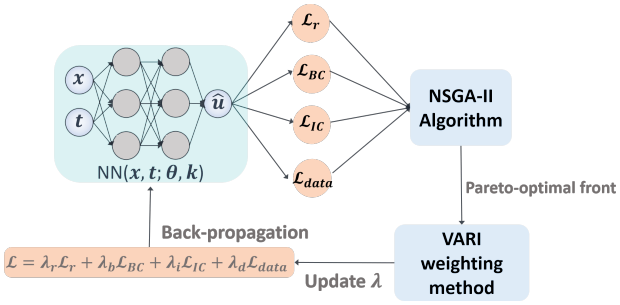


Figure 1: Schematic of multi-objective-VARI framework for PINN training.

Pareto Front Search for PINN Training

We first propose a multi-objective optimization framework for PINNs by employing the NSGA-II algorithm to search for the Pareto front considering the loss terms in Eq. 2. The NSGA-II algorithm provides a systematic and efficient strategy to address MOO problems with its population-based evolutionary framework and the ability to explore trade-offs systematically (Deb et al. 2002). By integrating NSGA-II into the PINN training process, our goal is to explore the parameter space of PINN and identify a Pareto front that optimally balances the different loss terms (i.e., $\mathcal{L}_r$, $\mathcal{L}_{BC}$, $\mathcal{L}_{IC}$, and $\mathcal{L}_{data}$), which will be used to determine the loss weights in the next section.

The integration of NSGA-II into the PINN training procedure is outlined in Algorithm 1. Specifically, the algorithm begins by initializing a population of candidate solutions P_0, where each individual represents a unique set of parameters θ and unknown parameters k in the PINN model. The initial populations are randomly sampled from the feasible space in the original NSGA-II algorithm. In this work, we fix half of the initial population with the current θ and k in the PINN model, while the remaining half of the population are randomly generated within feasible bounds ($\theta_{max}, \theta_{min}, k_{max}, k_{min}$). This population initialization method ensures that the algorithm explores a diverse range of solutions while still leveraging the information from the previous training results of the PINN.

Next, at each generation g, the algorithm evaluates the fitness of each individual in the population. At this stage, it should be noted that the neural network parameters θ^i and unknown parameters k^i of the ith individual should be first extracted from the population, and then assigned to the PINN model. Then the fitness of each individual is evaluated by calculating the four loss terms in Eq. 2 separately using the PINN model. These losses form the objective vector F^i for each individual, explicitly quantifying how well each candidate set of parameters satisfies the PDE, boundary and initial conditions, and observational data. Following evaluation, the population P_g undergoes non-dominated sorting, where individuals are ranked into Pareto fronts based on their performances across all loss terms. Next, parents for generating offspring are selected based on both their Pareto rank and crowding distances, using a crowded-comparison operator. The selected parents are then subjected to crossover and mutation operations to produce offspring Q_g, which are combined with the current population P_g to form a new population P_{g+1}. Finally, the algorithm terminates after a N_g generations, and the final population P_{N_g} contains a set of non-dominated solutions that approximate the Pareto front of the multi-objective optimization problem.

The proposed MOO-PINN framework uses MOO techniques to treat every competing loss term in the PINN loss function as a unique objective to be explored, which offers insight that scalarisation-based PINNs cannot provide. Specifically, the Pareto front returned by NSGA-II provides engineers with explicit trade-off visualizations, which quantify the marginal cost of tightening the residual error versus gaining a better data fit. This visual diagnostic allows engineers to flexibly select and design weight selection methods that are more in line with practical problems. Additionally, because the Pareto set is regenerated periodically during training, subsequent decisions about weight updates are therefore informed by the current frontier of attainable compromises, rather than relying on preset annealing curves or manual rules. Furthermore, the new directions supplied by the Pareto front can guide the Adam optimizer to escape local minima, which is particularly beneficial for inverse problems, where the parameter space is highly non-convex.

Algorithm 1: NSGA-II-Based Pareto Front Search for PINN

Input: population size N_p, number of generations N_g, current neural network parameters θ, unknwon parameters k, upper and lower bounds of the network parameters space θ_{max} and θ_{min}, unknown parameters range k_{max} and k_{min}

for $g = 0$ **to** N_g **do**
  for $i = 1$ **to** N_p **do**
    Extract the parameters θ^i and k^i from the population for the ith individual
    Assign θ^i and k^i to the PINN model
    Evaluate the multiple objectives using the PINN model:
    $F^i = [\mathcal{L}_r(\theta^i,k^i), \mathcal{L}_{BC}(\theta^i,k^i), \mathcal{L}_{IC}(\theta^i,k^i), \mathcal{L}_{data}(\theta^i,k^i)]$
  end
  Perform non-dominated sorting on P_g and assign Pareto ranks
  Compute crowding distances within each Pareto front of P_g
  Select parents using crowded-comparison operator based on Pareto rank and crowding distance
  Generate offspring population Q_g using crossover and mutation operators
  Combine parent and offspring populations to form a new populations: $P_{g+1} = P_g \cup Q_g$
end

Output: Pareto front of objective matrix F from the final population P_{N_g}

Variance-Aware Relative Improvement (VARI) Weighting Method

After obtaining the Pareto-optimal front through Algorithm 1, the next step is determining an optimal weighting strategy for the loss terms in the PINN training process. Drawing inspiration from existing weighting methods, we propose a novel weighting method called variance-aware relative improvement (VARI), designed explicitly for PINN training. The VARI method leverages insights gained from the variance of Pareto-optimal solutions, while also considering the relative improvement of each loss term over the training process. Specifically, the VARI method operates in two main stages: quantifying the relative improvement of objectives and computing their standard deviation across the Pareto front. The detailed steps of the VARI method are illustrated in Algorithm 2 and can be described as follows.

First, VARI quantifies the relative improvement of each loss term compared to its historical training performance. To achieve this, we define the historical baseline for each loss term $\mathcal{L}_j^{pre}$ as the minimum loss value observed over a specified historical window:

$$\mathcal{L}_j^{pre} = \min_{t \in T} \mathcal{L}_{t,j}^{hist}, \text{ where } T = \{t \mid t = T_{\text{end}} - \alpha, \ldots, T_{\text{end}}\} \tag{4}$$

where $\mathcal{L}_{t,j}^{hist}$ denotes the historical loss value of the jth objective recorded at training iteration t, T_{end} denotes the current training iteration, and α defines the length of the historical time window. It should be noted that due to the complex training dynamics of PINN, individual loss terms often exhibit significant fluctuations rather than monotonically or smoothly decreasing over time. Directly using the loss value from the previous training epoch as $\mathcal{L}_j^{pre}$ can thus lead to misleading assessments of relative improvement. Therefore, we use the minimum loss value observed across the last α training epochs to provide a stable benchmark for measuring subsequent relative improvements. Next, we calculate the mean performance of each objective across all current Pareto-optimal solutions:

$$\overline{F}_j = \frac{1}{N_p} \sum_{i=1}^{N_p} F_{ij} \quad i \in \{1, 2, \ldots, N_p\}, j \in \{1, 2, \ldots, M\} \tag{5}$$

where F_{ij} represents the jth objective value of the ith solution on the Pareto front. N_p represents the total number of Pareto solutions, and M is the number of objectives. The relative improvement ratio r_j for each objective is then defined as the ratio between the mean objective value across Pareto-optimal solutions and the historical baseline (i.e., $r_j = \frac{\overline{F}_j}{\mathcal{L}_j^{pre}}$). A lower relative improvement ratio r_j indicates significant improvement in the solutions obtained through the MOO process compared to previous training steps. This suggests that the corresponding objective has considerable room for further optimization in subsequent training and should thus be prioritized to help the PINN escape local optima. Conversely, a higher r_j implies slower improvement, indicating the corresponding objective may have reached a performance plateau within the current solution space and thus might not require immediate emphasis.

In the second stage, the VARI method evaluates the variability of Pareto solutions for each objective. This stage begins with normalizing the Pareto-optimal objectives to ensure fair comparison across different scales. Each objective F_{ij} is normalized by applying the max-min normalization :

$$F_{ij}^{norm} = \frac{F_{ij} - \min_{i \in N_p} F_{ij}}{\max_{i \in N_p} F_{ij} - \min_{i \in N_p} F_{ij}} \tag{6}$$

The standard deviation σ_j of each normalized objective across the Pareto front is then calculated as:

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{N_p} \left(F_{ij}^{norm} - \overline{F}_j^{norm}\right)^2}{N_p}} \quad j \in \{1, 2, \ldots, M\} \tag{7}$$

where $\overline{F}_j^{norm} = \frac{1}{N_p} \sum_{i=1}^{N_p} \overline{F}_{ij}^{norm}$ is the arithmetic mean of values of jth normalized objective. Finally, the composite score s_j for each objective is defined by integrating the relative improvement ratio and standard deviation, which are then used to calculate the weight for each loss term:

$$s_j = \frac{\sigma_j}{r_j}, \quad \lambda_j = \frac{M \cdot \exp(s_j / \mathcal{T}_j)}{\sum_{j=1}^{M} \exp(s_j / \mathcal{T}_j)} \tag{8}$$

where the score s_j intuitively balances the sensitivity of each objective and the recent trajectory of improvement, thus enabling adaptive prioritization of objectives based on evolving training dynamics and uncertainties. To convert these composite scores into explicit weights, we employ a temperature-controlled softmax normalization. The temperature parameter $\mathcal{T}$ controls the sensitivity of the softmax function. A smaller temperature value intensifies the differences among objectives, enabling sharper prioritization of those requiring immediate attention, such as data-driven losses in early training phases. Conversely, a higher temperature results in smoother weight adjustments, suitable for later training stages when objectives approach convergence. Considering that static weighting approach may incorrectly emphasize physics-driven terms too early under parameter uncertainty, slowing down convergence and reducing predictive accuracy, $\mathcal{T}_j$ is adaptively adjusted based on the current magnitude of the $\mathcal{L}_{data}$. Specifically, when the data-driven loss term $\mathcal{L}_{data}$ exceeds a predefined threshold ϵ, indicating insufficient parameter approximation, a smaller temperature (e.g., $\mathcal{T}_{data} = 0.5$) is allocated for this objective. On the other hand, when $\mathcal{L}_{data}$ reduces below the threshold, a uniform higher temperature (i.e., $\mathcal{T} = 1$) is applied across all objectives to ensure stable weight evolution.

The overall multi-objective PINN training framework is finally constructed in Algorithm 3. Specifically, after each cycle of MOO, the calculated weights λ from the VARI method are integrated back into the PINN training loop for standard gradient-based optimizers (e.g., Adam) to iteratively minimize the total weighted loss $\mathcal{L}(\theta, k)$. Rather than invoking NSGA-II and VARI at every epoch, the framework activates these steps only once every f_{MOO} epochs. This is because running only a single epoch of Adam between successive MOO and VARI steps may not allow the optimizer to sufficiently explore the parameter space under the latest set of weights. Additionally, updating weights frequently may disturb the momentum terms in Adam, jeopardizing the stability of its parameter updates. Finally, running MOO at each

Algorithm 2: Variance-Aware Relative Improvement (VARI) Weighting Method

Input: Pareto-optimal objective matrix $F \in \mathbb{R}^{N_P \times M}$, historical loss values $\mathcal{L}_{t,j}^{hist}$, historical time window α, threshold of data-driven loss term $\epsilon = 0.001$, and temperature vector $\mathcal{T}_j = 1$

Extract the historical baseline: $\mathcal{L}_j^{pre} \leftarrow \mathcal{L}^{hist}, \alpha$

Compute the mean objective values: $\overline{F}_j \leftarrow F_{ij}$

Calculate the relative improvement ratio: $r_j \leftarrow \overline{F_j}, \mathcal{L}_j^{pre}$

Normalize the objective matrix: $F_{ij}^{norm} \leftarrow F_{ij}$

Calculate standard deviation for normalized objectives: $\sigma_j \leftarrow F_{ij}^{norm}, \overline{F}_j^{norm}$

Compute the composite score: $s_i \leftarrow \sigma_i, r_i$

if $\mathcal{L}_{data} > \epsilon$ **then** $\mathcal{T}_{data} = 0.5$;

Computer the adaptive weights: $\lambda_j \leftarrow s_j, \mathcal{T}_j$

Output: Adaptive weights vector $\lambda \in \mathbb{R}^M$

epoch would impose an extra computational cost. Therefore, an intermittent activation schedule can achieve a balance between exploration, numerical stability, and computational efficiency. By applying MOO to capture the latest trade-offs between different loss terms, the proposed framework can handle complex loss landscapes that arise when additional regularization terms increase. Moreover, instead of using gradient statistics to update the loss weights, the proposed framework uses changes in the loss values and the dispersion of Pareto-optimal solutions to guide the optimizer continually adjusts its search direction.

Remark 1. It should be noted that the proposed MOO-VARI framework is not limited to a particular MOO algorithm or inner optimizer. Our goal is to cast PINN training in a multi-objective perspective and systematically allocate weights across competing loss terms. In addition to NSGA-II, one may use other evolutionary algorithms, gradient-based MOO, or surrogate MOO when objective evaluation is expensive. We also compare alternative MOO solvers (e.g., NSGA-III (Jain and Deb 2013), and Reference vector Guided Evolutionary Algorithm (RVEA) (Cheng et al. 2016)) in the experiments to show this interchangeability. Similarly, the update of neural network parameters does not depend on Adam. Other optimizers such as SGD, AdamW, or L-BFGS can be used without changing the framework.

Remark 2. The key contribution of this work lies in reframing the design of PINN loss functions through the lens of MOO, rather than developing novel optimization algorithms. In contrast to existing methods that dynamically adjust weights based on gradient or heuristics, our approach treats each loss term as an objective and optimizes them jointly to be equally good. This is a fundamental advantage of MOO: it inherently seeks balanced trade-offs across all objectives, avoiding overfitting to dominant terms and reducing dependence on expert knowledge or manual tuning.

Algorithm 3: Multi-objective-VARI PINN training framework

Input: Initial weights $\lambda \in \mathbb{R}^M$, number of training epochs N_{epochs}, hyperparameters of Algorithm 1 and Algorithm 2, and activation frequency of MOO f_{MOO}

for $t = 0$ **to** N_{epochs} **do**

  Integrate λ into the PINN overall loss function: $\mathcal{L}_(\theta, k) \leftarrow \sum_{j=1}^{M} \lambda_j \mathcal{L}_j(\theta, k)$

  Update the parameters of PINN by minimizing $\mathcal{L}_(\theta, k)$ via a Adam optimizer: $\theta, k \leftarrow$ Adam

  Store the current loss value in the historical loss list: $\mathcal{L}^{hist} \leftarrow \mathcal{L}_j$

  if $t \bmod f_{MOO} = 0$ **then**

    Perform MOO to find the Pareto-optimal front: $F \leftarrow Algorithm\ 1(\theta, k)$

    Compute the adaptive loss weights by VARI method: $\lambda \leftarrow Algorithm\ 2(F, \mathcal{L}^{hist})$

  end

end

Output: Trained PINN parameters (θ, k)

Experiments and Results

In this section, we illustrate the effectiveness of the proposed MOO-VARI framework by applying it to both PDE and ODE systems: Burgers equation, Lorenz system, and chemical process. We also compare our method with three other strategies: a novel loss balancing method called Relative Loss Balancing with Random Lookback (RelobraLo) (Bischof and Kraus 2025), a MOO-PINN framework that obtains the Pareto-optimal front and assigns weights with the CRITIC weighting method (denoted as MOO-CRITIC PINN) (Diakoulaki, Mavrotas, and Papayannakis 1995), and a baseline PINN with fixed weights. The detailed experimental settings of the case studies are provided in Appendix.

Burgers Equation. We consider the Burgers equation with Dirichlet boundary conditions as follows:

$$\begin{array}{ll} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, & x \in [-1, 1], t \in [0, 1], \\ u(x, 0) = -\sin(\pi x), & u(-1, t) = u(1, t) = 0 \end{array} \tag{9}$$

where the viscosity ν is an unknown parameter to be estimated during the training process of PINNs.

Fig. 2 presents the comparison between the predicted solutions by MOO-VARI PINN model and the reference solution, which demonstrates that MOO-VARI PINN can effectively solve Burgers equation. As shown in Fig. 3a, it is apparent that all models exhibit a rapid decrease in test loss during the initial training epochs. However, after the activation of their respective weighting methods at 1000 epochs, MOO-VARI PINN demonstrates superior convergence speed and achieves the lowest test loss by the end of training, indicating better predictive accuracy compared to the baseline, RelobraLo, and MOO-CRITIC methods. Additionally, Fig. 3b illustrates the approximation trajectory of

the viscosity parameter ν. The MOO-VARI PINN consistently achieves the most accurate estimation of the true parameters, demonstrating its capability to adaptively balance competing objectives and efficiently capture the true underlying physics of the system. These results indicate the effectiveness of our proposed framework in solving PDE systems.

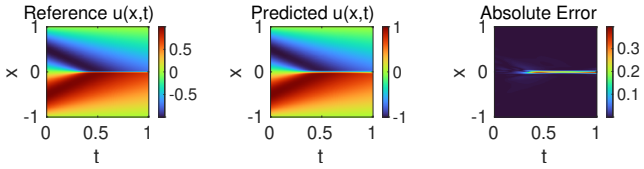
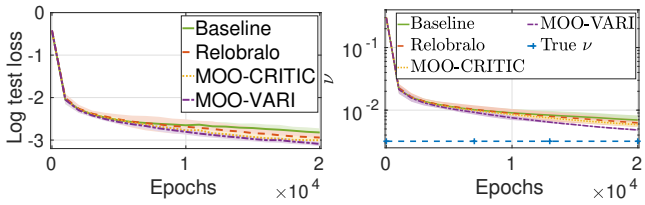


Figure 2: Burgers equation: Comparing the reference solution and predictions of MOO-VARI PINN.



(a) (b)

Figure 3: Burgers equations: (a) Evolution of the test loss during the training process. (b) Approximation of the true parameter value ν using different models.

Lorenz System. We consider the Lorenz system as follows:

$$\frac{dx}{dt} = \delta(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z \tag{10}$$

where x, y, z are the state variables. δ, ρ, and β are the three unknown parameters to be estimated.

Fig. 4 shows the reference solutions and predictions made by baseline PINN and MOO-VARI PINN at 10000, 30000 and 60000 epochs during the training process. It should be noted that the adaptive weighting mechanism of MOO-VARI PINN is activated after 10000 epochs. As training progresses to 30000 epochs, the predictions of baseline PINN remain inaccurate in certain critical portions of the trajectory. In contrast, the predictions of MOO-VARI PINN already demonstrate closer alignment with the reference solution, reflecting its ability to adaptively assign more suitable weights and thus achieve faster convergence. At the final training epochs, while both methods eventually converge closer to the reference solution, the MOO-VARI PINN achieves a consistently better overall match.

As shown in Fig. 5a, distinct differences among the various PINN models become apparent following the activation of their respective adaptive weighting schemes at 10,000 epochs. Remarkably, MOO-VARI PINN consistently outperforms all other methods throughout the training process, maintaining a notably faster convergence rate. Its test loss quickly decreases to 10^{-3} within approximately 20000 epochs and reaches 10^{-4} earlier than other methods. Given the available data and collocation points, this level of precision is considered close to the practical limit. Additionally, Fig. 5b shows that different MOO solvers deliver similar gains in convergence speed and accuracy, which confirm the generality of the proposed framework. As depicted in Fig. 6, the MOO-VARI PINN model shows superior performance for parameter identification among all methods, which has the fastest convergence speed toward the true parameter values. The result highlights that the VARI weighting scheme effectively prioritizes critical objectives at each training stage by explicitly incorporating information of Pareto-optimal front and relative improvement metrics, leading to superior convergence behavior and model accuracy.

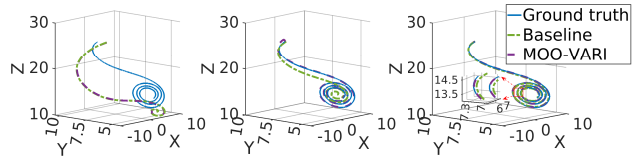
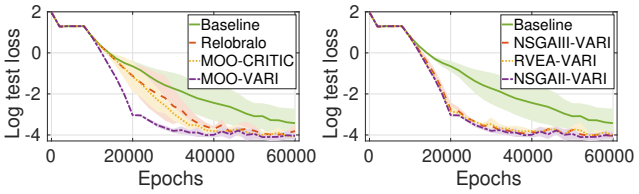


Figure 4: Lorenz system: Comparison of the reference solution and predictions of baseline PINN and MOO-VARI PINN at 10000 (left), 30000 (middle) and 60000 (right) epochs during training.



(a) (b)

Figure 5: Lorenz system: (a) Evolution of the test loss during the training process. (b) Evolution of the test loss obtained with different MOO algorithm coupled with VARI.

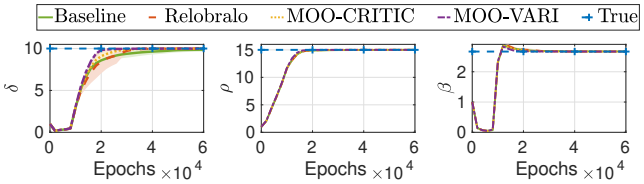


Figure 6: Lorenz system: Approximation of the true parameter value δ, ρ, β.

Application to a Nonlinear Chemical Process. In process systems modeling, the PINN formulation is frequently generalized to physics-informed recurrent neural network (PIRNN), whose internal state update naturally accommodates time-series data streams while still embedding first-principles constraints for engineering applications. This section demonstrates the effectiveness of the proposed

Case study	Baseline PINN	MOO-VARI PINN	One-time MOO-VARI
Burgers equation	6 min 16 s	6 min 30 s	11 s
Lorenz system	7 min 56 s	9 min 22 s	12 s
CSTR	33 min 27 s	40 min 45 s	183 s

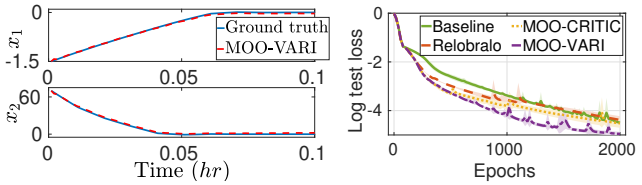
Table 1: Computational time for training different case studies.

MOO-VARI framework using a continuous stirred tank reactor (CSTR). The system dynamics can be presented as follows (Heidarinejad, Liu, and Christofides 2012):

$$\begin{aligned} \frac{dC_A}{dt} &= \frac{F_{in}}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{R_c T}} C_A^2 \\ \frac{dT}{dt} &= \frac{F_{in}}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{R_c T}} C_A^2 + \frac{Q_h}{\rho_L C_p V} \end{aligned} \tag{11}$$

where C_A and T are two state variables. Our goal is to use PIRNN model to capture the dynamic behavior of the reactor so that it can be used for prediction and operation of CSTR. The inputs and states of the system are formulated as $u^T = [\Delta C_{A0} \quad \Delta Q_h]$ and $x^T = [C_A - C_{As} \quad T - T_s]$, such that the equilibrium point (C_{As}, T_s) of the system is at the origin of the state-space. F_{in} and k_0 are assumed to be unknown and will be approximated during the training process.

Fig 7a indicates that the state profiles predicted by the MOO-VARI PIRNN model are in line with the ground-truth CSTR response, which demonstrates that MOO-VARI method can achieve reliable predictive performance. As shown in Fig. 7b, all models share identical fixed weights during the first 100 epochs. It can be seen that the MOO-VARI PIRNN displays the steepest drop in test loss, rapidly outperforming other models and ultimately attaining the lowest error level. As shown in Fig. 8, parameter estimation is particularly challenging in this case because k_0 is of the order 10^6. Even small deviations in its estimate can alter the reaction rate and thus destabilize training. Similarly, the MOO-VARI PIRNN achieves the fastest and most accurate convergence for both parameters, approaching the true values within the 1500 epochs and then remaining stable.



(a) (b)

Figure 7: CSTR: (a) The state trajectories predicted by MOO-VARI PIRNN with the initial condition $(-1.5\,\mathrm{kmol/m^3}, 70\,\mathrm{K})$. (b) Evolution of the test loss during the training process.

Computational Resources. Table 1 compares the computational times required by the baseline PINN and the proposed MOO-VARI framework for the three case studies. Although the integration of the MOO framework into the training process results in slightly increased training time,

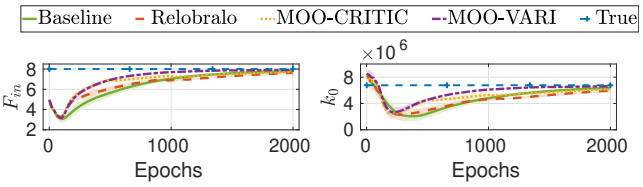


Figure 8: CSTR: Parameter estimation of F_{in} and k_0.

the extra cost is moderate and manageable. The extra cost per MOO-VARI activation mainly comes from evaluating each individual's objectives in NSGA-II, which scales with the number of objectives M, the population size N_p, and the complexity of the neural network architecture. In PINN framework, M is fixed. Therefore, the extra cost is influenced by N_p and network complexity, while only a small N_p is typically sufficient. Specifically, each activation of the MOO-VARI framework (e.g., 11 s for Burgers' equation) accounts for only a minor portion of the overall training time, demonstrating that the improvement of accuracy and convergence speed of the MOO-VARI framework is achieved with a small additional computational burden. The detailed cost scale of NSGA-II algorithm has been discussed in (Doerr and Qu 2023). Additionally, the MOO procedure is readily parallelized using distributed computing architectures, which allows the evaluation of the Pareto front and the corresponding weight adjustment to be performed simultaneously on distributed nodes or GPU accelerators. Specifically, distributing the population across two nodes reduces the single activation time from 11 s to 5 s for the Burgers equation, from 12 s to 6 s for the Lorenz system, and from 183 s to 72 s for the CSTR.

Conclusion

In this work, we proposed an adaptive weighting for physics-informed machine learning to dynamically adjust the weights of each loss term, especially for inverse problems with parameter uncertainty. Specifically, the MOO technique using NSGA-II was integrated into the training procedure of PINNs to explore Pareto-optimal solutions. Additionally, a novel VARI weighting method was introduced to convert the rich Pareto-front information into adaptive loss weights, explicitly accounting for both objective sensitivity and historical training trajectories. Compared to the baseline PINN, as well as other state-of-the-art weighting approaches such as RelobraLo and CRITIC methods, the proposed MOO-VARI PINN exhibited superior convergence speeds, lower test errors, and more accurate parameter identification results.

Acknowledgments

This study was supported by A*STAR MTC YIRG 2022 Grant (M22K3c0093) and NRF-CRP Grant 27-2021-0001.

References

Bischof, R.; and Kraus, M. A. 2025. Multi-objective loss balancing for physics-informed deep learning. *Computer Methods in Applied Mechanics and Engineering*, 439: 117914.

Cai, S.; Mao, Z.; Wang, Z.; Yin, M.; and Karniadakis, G. E. 2021a. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12): 1727–1738.

Cai, S.; Wang, Z.; Wang, S.; Perdikaris, P.; and Karniadakis, G. E. 2021b. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6): 060801.

Cheng, R.; Jin, Y.; Olhofer, M.; and Sendhoff, B. 2016. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE transactions on evolutionary computation*, 20(5): 773–791.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2): 182–197.

Diakoulaki, D.; Mavrotas, G.; and Papayannakis, L. 1995. Determining objective weights in multiple criteria problems: The critic method. *Computers & Operations Research*, 22(7): 763–770.

Doerr, B.; and Qu, Z. 2023. From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 12408–12416.

Elhamod, M.; Bu, J.; Singh, C.; Redell, M.; Ghosh, A.; Podolskiy, V.; Lee, W.-C.; and Karpatne, A. 2022. CoPhy-PGNN: Learning physics-guided neural networks with competing loss functions for solving eigenvalue problems. *ACM Transactions on Intelligent Systems and Technology*, 13(6): 1–23.

Gunantara, N. 2018. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, 5(1): 1502242.

Hao, Z.; Yao, J.; Su, C.; Su, H.; Wang, Z.; Lu, F.; Xia, Z.; Zhang, Y.; Liu, S.; Lu, L.; and Zhu, J. 2024. PINNacle: A comprehensive benchmark of physics-informed neural networks for solving PDEs. In *Advances in the 38th Neural Information Processing Systems*, 37: 76721–76774.

Heidarinejad, M.; Liu, J.; and Christofides, P. D. 2012. Economic model predictive control of nonlinear process systems using Lyapunov techniques. *AIChE Journal*, 58(3): 855–870.

Hwang, Y.; and Lim, D. 2024. Dual cone gradient descent for training physics-informed neural networks. In *Advances in the 38th Neural Information Processing Systems*, 37: 98563–98595.

Jain, H.; and Deb, K. 2013. An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on evolutionary computation*, 18(4): 602–622.

Ji, W.; Qiu, W.; Shi, Z.; Pan, S.; and Deng, S. 2021. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 125(36): 8098–8106.

Karniadakis, G. E.; Kevrekidis, I. G.; Lu, L.; Perdikaris, P.; Wang, S.; and Yang, L. 2021. Physics-informed machine learning. *Nature Reviews Physics*, 3(6): 422–440.

Krishnapriyan, A.; Gholami, A.; Zhe, S.; Kirby, R.; and Mahoney, M. W. 2021. Characterizing possible failure modes in physics-informed neural networks. In *Advances in the 35th Neural Information Processing Systems*, 34: 26548–26560.

Li, L.; Liu, F.; and Li, C. 2014. Customer satisfaction evaluation method for customized product development using Entropy weight and Analytic Hierarchy Process. *Computers & Industrial Engineering*, 77: 80–87.

Liu, Y.; Cai, L.; Chen, Y.; Ma, P.; and Zhong, Q. 2024. Variable separated physics-informed neural networks based on adaptive weighted loss functions for blood flow model. *Computers & Mathematics with Applications*, 153: 108–122.

Matsubara, T.; and Yaguchi, T. 2025. Number Theoretic Accelerated Learning of Physics-Informed Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(1): 595–603.

McClenny, L. D.; and Braga-Neto, U. M. 2023. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474: 111722.

Mirjalili, S.; Jangir, P.; and Saremi, S. 2017. Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. *Applied intelligence*, 46(1): 79–95.

Misyris, G. S.; Venzke, A.; and Chatzivasileiadis, S. 2020. Physics-informed neural networks for power systems. In *2020 IEEE power & energy society general meeting (PESGM)*, 1–5. IEEE.

Müller, J.; and Zeinhofer, M. 2023. Achieving high accuracy with PINNs via energy natural gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, 202: 25471–25485. PMLR.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378: 686–707.

Rathore, P.; Lei, W.; Frangella, Z.; Lu, L.; and Udell, M. 2024. Challenges in Training PINNs: A Loss Landscape Perspective. In *Proceedings of the 41st International Conference on Machine Learning*, 235: 42159–42191. PMLR.

Sharma, S.; and Kumar, V. 2022. A comprehensive review on multi-objective optimization techniques: Past, present and future. *Archives of Computational Methods in Engineering*, 29(7): 5605–5633.

Sun, L.; Zhang, Z.; Wang, Z.; Wang, Y.; Wan, Q.; Li, H.; Peng, H.; and Yu, P. S. 2025. Pioneer: Physics-informed Riemannian Graph ODE for Entropy-increasing Dynamics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(12): 12586–12594.

Van Der Meer, R.; Oosterlee, C. W.; and Borovykh, A. 2022. Optimally weighted loss functions for solving pdes with neural networks. *Journal of Computational and Applied Mathematics*, 405: 113887.

Wang, F.; Zhai, Z.; Zhao, Z.; Di, Y.; and Chen, X. 2024. Physics-informed neural network for lithium-ion battery degradation stable modeling and prognosis. *Nature Communications*, 15(1): 4332.

Wang, S.; Teng, Y.; and Perdikaris, P. 2021. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5): A3055–A3081.

Wang, S.; Yu, X.; and Perdikaris, P. 2022. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449: 110768.

Wang, Z.; Parhi, S. S.; Rangaiah, G. P.; and Jana, A. K. 2020. Analysis of weighting and selection methods for pareto-optimal solutions of multiobjective optimization in chemical engineering applications. *Industrial & Engineering Chemistry Research*, 59(33): 14850–14867.

Xiang, Z.; Peng, W.; Liu, X.; and Yao, W. 2022. Self-adaptive loss balanced physics-informed neural networks. *Neurocomputing*, 496: 11–34.

Yang, Z.; Qiu, Z.; and Fu, D. 2023. DMIS: Dynamic mesh-based importance sampling for training physics-informed neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4): 5375–5383.

Ye, Y.; Can, C.; Christopher, P.; and Xue, L. 2025. ParetoFlow: Guided Flows in Multi-Objective Optimization. In *Proceedings of the 30th International Conference on Learning Representations*, 72594–72620.

Ye, Y.; Li, L.; Lin, Q.; Wong, K.-C.; Li, J.; and Ming, Z. 2022. Knowledge guided Bayesian classification for dynamic multi-objective optimization. *Knowledge-Based Systems*, 250: 109173.

Yusop, Z. B.; Ahmed, K.; Shirazi, S. M.; and Zardari, N. H. 2015. *Weighting methods and their effects on multi-criteria decision making model outcomes in water resources management*. Springer.

Zobeiry, N.; and Humfeld, K. D. 2021. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101: 104232.