Learning to Predict Pareto-optimal Solutions From Pseudo-weights

Kalyanmoy Deb[0000−0001−7402−9939], Aryan Gondkar, and Suresh Anirudh

Computational Optimization and Innovation Laboratory, Michigan State University, East Lansing, MI 48824, USA
{kdeb,gondkara,suresha2}@msu.edu
https://www.coin-lab.org

COIN Report Number 2022010

Abstract. Evolutionary Multi-objective optimization (EMO) algorithms attempt to find a well-converged and well-diversified set close to true Pareto-optimal solutions. However, due to stochasticity involved in EMO algorithms, the uniformity in distribution of solutions cannot be guaranteed. Moreover, the follow-up decision-making activities may demand finding more solutions in specific regions on the Pareto-optimal front which may not be well-represented by the obtained EMO solutions. In this paper, we train machine learning algorithms to capture the relationship between pseudo-weight vectors, derived from location of EMO-obtained non-dominated objective vectors, and their respective decision variable vectors. The learnt system can then be utilized to predict the corresponding variable vector for any desired pseudo-weight vector. The proposed methodology is applied to a number of problem instances to demonstrate its working and usefulness in arriving at a desired distribution of near Pareto-optimal solutions. The methodology has the potential to be embedded within an EMO algorithm to produce a better distributed set of solutions, check the validity of apparent gaps in obtained fronts, and also to help find more non-dominated solutions at the preferred regions of the Pareto-optimal front for effective decision-making purposes.

Keywords: deep neural networks · multi-objective optimization · pseudo-weights · multi-criterion decision-making.

1 Introduction

Evolutionary multi-objective optimization (EMO) algorithms are capable of finding multiple trade-off solutions near or on the Pareto-optimal (PO) front [5,2] for multi-objective optimization problems having more than one objective functions and multiple constraint functions. The algorithms have been extended to handle more than three-objective problems – known as many-objective optimization (MaO) problems [15,3,10]. However, EMO or EMaO algorithms are stochastic in nature and despite a great deal of effort in finding a well-distributed set of

trade-off solutions, they often fail to find the desired distribution. Besides the stochasticity inherent in the algorithms, the complexity offered by a problem near the PO front is another reason for not arriving at a uniformly distributed set of trade-off solutions.

Moreover, the ensuing decision-making (DM) task in choosing a single preferred trade-off solution must focus on a specific region on the obtained trade-off solutions. It is not guaranteed that the EMO or EMaO-obtained trade-off solutions have enough solutions corresponding to the preferred region by the decision-makers. Often, the DM task cannot be pursued before the EMO or EMaO run is executed, as DMs may not be certain about the preference information without the knowledge of an optimized trade-off solution set.

Both the above scenarios demand that a computationally quick and simple procedure of creating a new trade-off solution easily at any part of the optimized front. This task is akin to various machine learning tasks in which input-output relationships are learnt from a set of training data and the learnt system is then used to predict output from a given and unseen input. In our task, the input-output training data are the EMO or EMaO-obtained trade-off solutions for which input is a unique indicator of a solution on the front and the output is the solution vector ($\mathbf{x}$). In this study, we use the pseudo-weight vector [5] as a unique indicator of a trade-off solution on the front. An advantage of using a pseudo-weight vector is that every component is normalized to lie in $[0, 1]$. After a machine learning system is learnt, any new pseudo-weight vector can be used as an input to find the respective $\mathbf{x}$-vector for which the corresponding $\mathbf{f}$-vector can be computed using the given objective functions.

In the remainder of this proof-of-principle study, we briefly outline a number of existing studies related to finding a uniformly distributed set of Pareto-optimal (PO) solutions, focusing on filling gaps in obtained fronts in Section 2. The proposed machine learning procedure for the post-optimality study is described in Section 3. Results on multi-objective and many-objective constrained and unconstrained test problems and engineering problems are presented in Section 4. Finally, Section 5 summarizes the findings of this study and proposes a number of viable extensions.

2 Existing Studies

Use of machine learning methods in EMO is well studied in recent literature. Machine learning methods are used for surrogate modeling ($\mathbf{x}$ to $\mathbf{f}$ mapping) [6], *innovization* ($\mathbf{x}$ to $\mathbf{x}$ mapping) [12], and as genetic operators (sampling in $\mathbf{x}$ space) [9].

Surrogate assisted optimization methods use machine learning methods as a computationally cheap alternative to expensive function evaluations. Exploitative optimization is performed using the surrogate function and high-fidelity evaluations are computed sparsely. Surrogate functions map from $\mathbf{x}$ to $\mathbf{f}$, thereby not helping our cause here. Inverse models [8] map $\mathbf{f}$ to $\mathbf{x}$, but require knowledge of true $\mathbf{f}$-vector to produce the requisite $\mathbf{x}$-vector, causing difficulties in DM

efforts. Our proposed method maps $\mathbf{w}$ to $\mathbf{x}$ (where $\mathbf{w}$ are pseudo-weights [5], which represent place-holder information of a solution in the PO front), making it convenient for DM purposes.

Machine learning methods have been used to aid EMO algorithms in achieving better convergence. Mittal et al [12] proposed a number of *innovization*-based progress operators that uses machine learning to learn meaningful search advances. These operators are based on intra-generational solutions that aid in diversity and convergence. He et al [9] proposed a Generative Adversarial Network (GAN) driven optimization algorithm, where a GAN is trained on current solutions and is then used to sample off-spring. In our proposed method, the machine learning models are conditioned on pseudo-weights and hence lead to a more targeted approach for finding non-dominated solutions directly.

Gaps often exist on obtained PO front. It is important for DMs to confirm if the gap really exists or is a shortcoming of the chosen EMO algorithm. If a gap truly exists, it becomes interesting for DMs to know what causes a gap and what properties of $\mathbf{x}$-vector enables a gap on a PO front. Pellicer et al [14] proposed a novel multi-step gap-finding method where converged solutions are clustered into gaps and a reference-direction based method [7] is used to validate lack of solutions in the gap. This method requires repeated runs of optimization algorithm, contrary to our proposed method where we do not need to optimize again to find new points in the apparent gaps.

3 Proposed Machine Learning Based EMO Procedure

An application of an EMO/EMaO algorithm can produce a set of H non-dominated (ND) solutions ($\mathbf{x}^{(k)} \in R^n$, for $k = 1, 2, \ldots, H$). Each of these solutions can then be evaluated to compute the respective objective vectors $\mathbf{f}^{(k)} \in R^M$, for $k = 1, 2, \ldots, H$. From the position of each objective vector on the entire ND set, the respective pseudo-weight vector $\mathbf{w}^{(k)} = \left(w_1^{(k)}, w_2^{(k)}, \ldots, w_M^{(k)}\right)$, ($\mathbf{w}^{(k)} \in R^M$) can be computed as follows:

$$w_i^{(k)} = \frac{\left(f_i^{\max} - f_i^{(k)}\right) \Big/ \left((f_i^{\max} - f_i^{\min}\right)}{\sum_{j=1}^{M} \left(f_j^{\max} - f_j^{(k)}\right) \Big/ \left((f_j^{\max} - f_j^{\min}\right)}. \tag{1}$$

The pseudo-weight vector for a ND solution can be viewed as a representative identity on the ND set. For a two-objective problem the best f_1 solution corresponds to a pseudo-vector of $(1, 0)$, meaning that the solution is 100% importance for f_1 and no importance for f_2.

Clearly, Equation 1 indicates that pseudo-weights are derived from the objective values of ND set. If the ND solutions are sorted according to objective vectors, the respective $\mathbf{w}$-vectors will also get sorted in a similar manner. The studies on "innovization" concept [4] have revealed Pareto-optimal (PO) solutions usually possess certain patterns or constancy with respect to certain variables. Thus, when the ND solutions are close enough to the true Pareto-optimal

(PO) set, it is expected that for practical problems the respective $\mathbf{x}$-vectors would be somehow related to the pseudo-weight vectors. This then motivates us to capture the relationships between the derived $\mathbf{w}$-vectors and respective $\mathbf{x}$-vectors of the ND set using a machine learning method. Once the patterns, if any, are captured, a new ND solution ($\bar{\mathbf{x}}$-vector) is expected to be found by using a desired $\bar{\mathbf{w}}$-vector.

Figure 1 illustrates the training and testing procedure for developing a machine learning method. The procedure is presented in steps below:

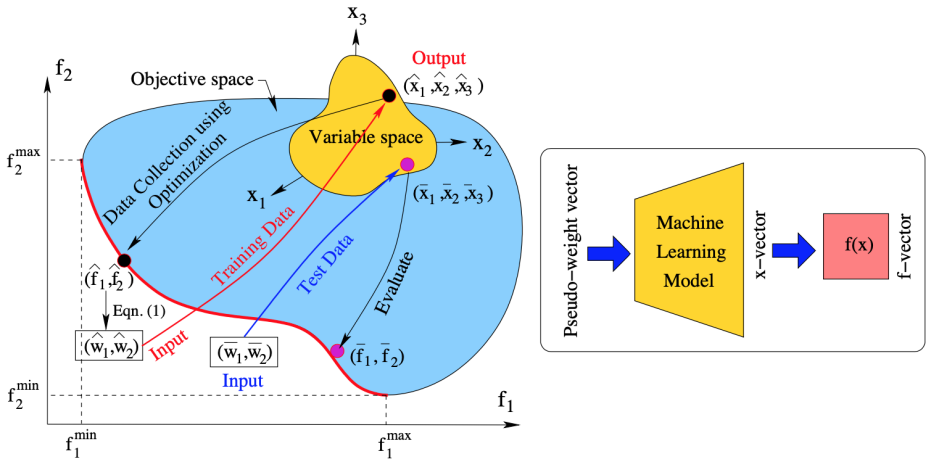


Fig. 1: Training data generation and test data to create new Pareto-optimal solution.

Step 1: From an EMO/EMaO-obtained ND set, calculate the pseudo-weight ($\mathbf{w}^{(k)}$)-vector for each solution (k) of the set using Equation 1.

Step 2: Prepare training data $(\mathbf{w}^{(k)}, \mathbf{x}^{(k)})$ for $k = 1, 2, \ldots, H$, with $\mathbf{w}^{(k)}$ as input and $\mathbf{x}^{(k)}$ as output.

Step 3: Train a ML method using the training data.

Step 4: Use the trained ML model to find $\bar{\mathbf{x}}$ for any specific desired $\bar{\mathbf{w}}$ vector. Then, compute $\bar{\mathbf{f}}$ from $\bar{\mathbf{x}}$.

The resulting trained model can be used for various purposes.

Task 1: It can be used to check the validity of the obtained trained model. Pseudo-weights can be chosen at random from the entire PO set for testing purposes. Care is taken to not choose a training data as a test data.

Task 2: It can also be used to find new ND solutions in the apparent gaps in the EMO/EMaO-obtained ND set. This can be achieved by creating pseudo-weights in the gaps of the pseudo-weight space and then creating ND points by the trained model.

Task 3: It can be used to evaluate if an apparent gap in EMO/EMaO-obtained ND set is truly a gap. This can be determined by first creating pseudo-weights in the gaps and then finding resulting $\mathbf{x}$ using the trained model and

then computing their **f** vectors to identify if they are dominated by the rest of the ND objective vectors.

Task 4: It can be used to quickly populate the ND set in a desired part of the ND set, mainly for decision-making or for a better visualization purposes. This can be achieved in two ways. First, creating suitable pseudo-weight vectors in the region of interest and then using the trained model to find new and hopefully non-dominated points.

Task 5: It can be used at intermediate generations of an EMO/EMaO run as an offspring creation mechanism. This can be achieved by first developing a trained model using the current ND points and then selecting pseudo-weights in less-dense areas of the pseudo-weight space and then using the model to create new solution.

In this study, we demonstrate the first four tasks here and leave the fifth task as an integral part of a new EMO algorithm requiring implementation and testing.

3.1 Training of Deep Neural Networks

Before we present results of our proposed method, we discuss machine learning methods considered in this study for developing the trained model. The problem here can be stated as a task of predicting an n-dimensional **x**-vector from an M-dimensional pseudo-weight (**w**) vector (where M is the number of objectives of the problem). Since usually $M \ll n$, the model development from a few input parameters to a large number of output parameters is a challenging task. We use two different machine learning methods for this purpose: (i) a deep neural network (DNN) approach and (ii) a Gaussian process regression (GPR) approach. In both DNN and GPR approaches, **x**-vectors were normalized to zero mean and unit variance as required by the model. Pseudo-weight vectors are already within [0,1] and are not normalized.

For every problem, the PO front is computed using NSGA-III [3], with a population size $N = 110M + 10$. These numbers are chosen with a trial-and-error study. Of these N PO points, $100M$ points are used as training points, and $10M$ points are taken as the test set. Finally, for the DNN, the remaining 10 points are used as a validation set, but discarded for GPR to maintain the similarity of the training data. Understanding the effect of size of dataset and other hyperparameters are left for future studies. Validation set was sampled uniformly from the training set for the purposes of model selection in the case of DNN. It is important to note that the proposed method is not conditional on the source of the training set. Hence, the PO front based training set can be replaced by the non-dominated set of an MOEA at the end of a particular generation.

Deep Neural Network (DNN) approach: Multi-layer perceptrons (MLPs) with pseudo-weights as inputs and variables (**x**) as outputs are implemented using the PyTorch [13] library and hyper-parameters are optimized using the Optuna software [1]. Owing to proof-of-concept nature of the study, DNNs with

1 to 6 hidden layers are used with ReLU activation and trained using Adam optimizer [11]. The complexity of the DNNs and granularity of hyper-parameters can be increased for handling more complex problems.

Gaussian Regression (GPR) Approach An approach similar to surrogate modeling is used for training GPR models. Every output, x_i is modeled independently. A sundry of kernels, mean functions and other hyper-parameters are considered for a grid-search for finding the most suitable setting.

3.2 Handling Variable Bounds and Constraints

In an optimization problem, the variable vectors are usually bounded within lower and upper bounds. Since a DNN or a GPR approach does not usually restrict its output values automatically within any bound, the resulting output values for a test input data can be out of bounds, if a proper care is not taken. In this study, we normalize the output values as needed by the ML model. The output value from the system is then de-normalized and clipped to within their specified lower and upper bounds.

Constraint satisfaction is also a strict requirement in an optimization task. The resulting $\mathbf{x}$-vectors from the trained model may not guarantee to satisfy all constraints automatically. In this study, we simply discount infeasible $\mathbf{x}$ solutions, in case such a solution is created by the trained model; but a more sophisticated constrained handling method can be used during the training process. For example, constraint value of each constraint can be included as additional output to the DNN or GPR process. During testing, if any $\mathbf{w}$-vector (input) produces a positive constraint value (meaning a constraint violation), the solution is simply ignored. In this case, some training data with positive constraint violation, but non-dominated to the feasible ND set must be used to achieve a better training process. In this proof-of-principle study, we do not use any such sophisticated method.

4 Results

First, we present results on two-objective unconstrained and constrained test problems. Thereafter, we show results on three-objective problems, followed by a few many-objective problems. For each experiment, 11 runs are performed and their mean results are presented.

Tables 1 - 4 show Mean Absolute Errors (MAE) of test set on multi- and many-objective problems with different modeling approaches. The mean $\mathbf{x}$ is the average MAE scaled based on variable bounds between true $\mathbf{x}$-vectors (from the PO set) and model-obtained $\mathbf{x}$-vectors for the $10M$ test pseudo-weight vectors. Similarly, the mean $\mathbf{f}$ is the average MAE scaled based on range of objectives on the Pareto front ($\mathbf{f}^{\text{nadir}} - \mathbf{f}^{\text{ideal}}$) between the true $\mathbf{f}$-vectors and the model-obtained $\mathbf{x}$-vectors for the test pseudo-weight vectors. Their standard deviation values across the PO set are also presented in the table. 'Random' signifies that test pseudo-weight data are chosen at random on the entire PO front (Task 1).

'Continuous' and 'Edge' signify that a continuous region of a gap or a gap at one of the extreme parts of the PO set (Task 2) is used.

4.1 Two-objective Problems

Table 1 presents prediction errors in **x** and **f**-vectors of test data by the DNN approach on two-objective unconstrained ZDT and constrained (BNH and OSY) problems. We can see that the error values are low for all the test problems.

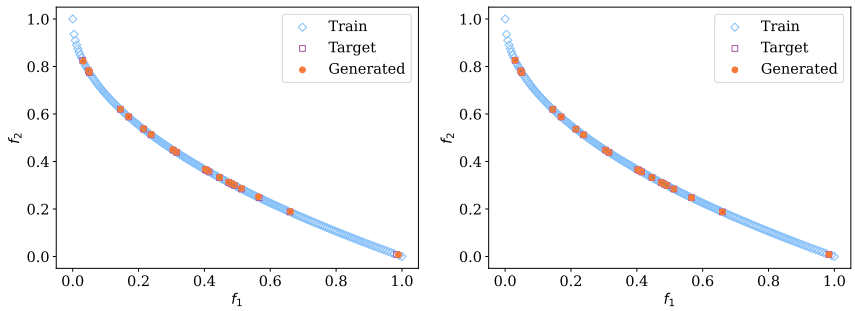
Table 1: Performance of DNN approach on two-objective problems.

Problem	Test Data	Mean **x**	Mean **f**	Std. Dev. in **x**	Std. Dev. in **f**
ZDT1	Continuous	6.281E-04	9.264E-03	1.051E-04	1.374E-03
	Edge	2.012E-03	1.864E-02	4.350E-04	4.332E-03
	Random	4.606E-04	6.918E-03	2.575E-04	3.689E-03
ZDT2	Continuous	7.387E-04	1.321E-02	1.242E-04	2.357E-03
	Edge	1.313E-03	3.986E-02	1.207E-04	3.680E-03
	Random	7.260E-04	1.387E-02	4.065E-04	6.786E-03
ZDT3	Continuous	1.026E-03	3.948E-02	2.030E-04	1.994E-02
	Random	4.619E-04	1.757E-02	2.591E-04	1.434E-02
BNH	Continuous	3.701E-03	2.760E-03	4.185E-03	5.378E-02
	Random	2.491E-03	1.407E-03	4.384E-03	7.347E-02
OSY	Continuous	2.172E-03	2.720E-02	3.685E-03	6.518E-01
	Random	7.833E-03	3.607E-02	3.323E-02	3.521E+00

Validating Task 1: Figure 2a shows that the randomly chosen pseudo-weight vectors produce **x**-vectors that make **f**-vectors fall on the PO front. The DNN-generated **f**-vectors (shown in red filled cirles) fall almost on top of the corresponding target **f**-vectors (shown with red open square). This is not an easy feat, as the DNN-learnt model produces **x**-vectors from supplied **w**-vectors and then the **x**-vectors are evaluated to compute **f**-vectors for plotting. This validates Task 1, proposed in Section 3 on ZDT1 problem. Note that the test data (red) are excluded from the training data (blue).

Next, we apply the GPR approach and results are tabulated in Table 2. Interestingly, much smaller error values are observed with GPR approach on the two-objective problems compared to the DNN approach. Figure 2b shows that GPR also can generate PO points very close to tagret **f**-vectors for the same set of pseudo-weights.

Validating Task 2: Next, we investigate if the proposed models can produce points on continuous gaps in the PO front produced by an EMO algorithm (Task 2). Outcome of DNN and GPR approaches are shown in the ZDT1 problem when the PO front has a gap on one of the extreme part of the PO front, as shown in Figure 3. Error values are shown in Tables 1 and 2. Only blue points are used to train the machine learning models, but red points are used to test.



(a) DNN approach (b) GPR approach

Fig. 2: DNN and GPR approaches on random test data for ZDT1 problem, demonstrating Task 1.

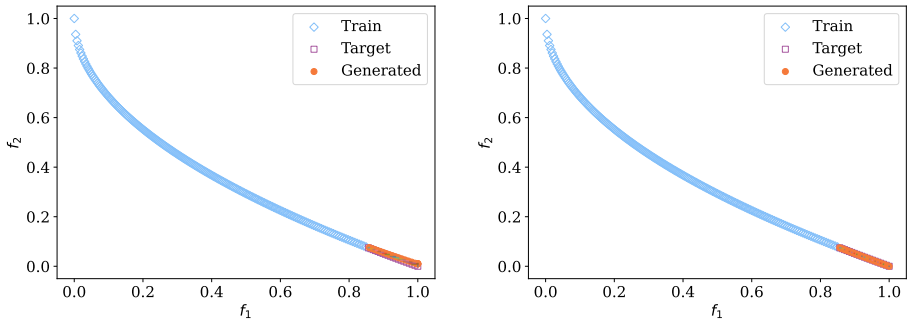
Table 2: Performance of GPR approach on two-objective problems.

Problem	Test Data	Mean **x**	Mean **f**	Std. Dev. in **x**	Std. Dev. in **f**
ZDT1	Continuous	2.625E-08	8.466E-08	5.694E-08	1.806E-07
	Edge	1.086E-06	2.447E-05	8.081E-07	1.820E-05
	Random	5.525E-09	3.462E-06	3.064E-09	7.511E-06
ZDT2	Continuous	1.083E-04	3.633E-03	3.903E-05	1.320E-03
	Edge	8.651E-04	3.811E-02	2.308E-04	1.023E-02
	Random	1.398E-05	4.200E-04	1.946E-05	6.093E-04
ZDT3	Continuous	3.182E-05	3.365E-03	9.706E-06	2.015E-03
	Random	3.659E-06	3.098E-04	4.985E-06	8.355E-04
BNH	Continuous	2.343E-03	1.274E-04	3.054E-03	2.224E-03
	Random	1.890E-03	2.073E-04	3.004E-03	9.670E-03
OSY	Continuous	1.471E-03	1.243E-03	4.562E-03	5.371E-02
	Random	1.491E-03	1.273E-03	1.146E-02	3.407E-01

This requires ML models to learn how to extrapolate learnt relationships from training to test data and is always a harder task. Also, note that the models had to learn (i) how to create a suitable **x**-vector so that the resulting **f**-vector falls on the PO front, and (ii) make the **f**-vector fall at a place congruent to the chosen pseudo-weight vectors.

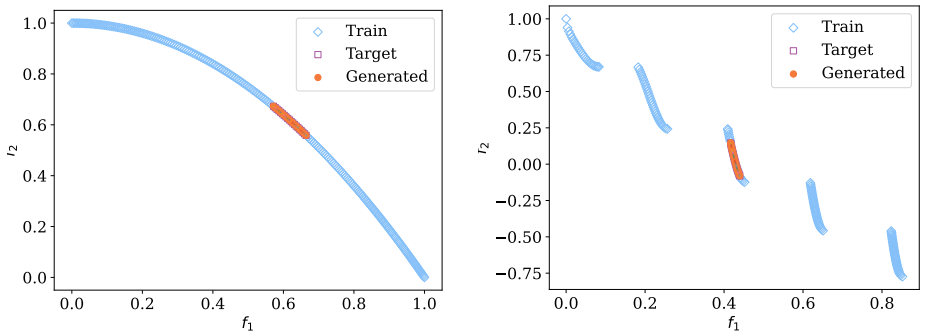
Figure 4 shows that GPR approach can also reproduce points at the middle part of the PO front on ZDT2 and ZDT3 problems. Here, too, training data did not include the points in the gap. Missing pseudo-weights at the gaps are estimated from the training data and are used to generate PO points in the gaps.

Validating Task 3: In ZDT3 problem, there are natural gaps in the PO front. Thus, there will be gaps in the pseudo-weight space when they are computed from a set of EMO-obtained PO points. Next, we investigate what our trained DNN and GPR models would produce, if pseudo-weights in the natural gaps are chosen to find the respective **x**-vectors. Towards this task (Task 3), we create



(a) DNN approach (b) GPR approach

Fig. 3: DNN and GPR approaches in finding edge gap points on ZDT1 problem, demonstrating Task 2.



(a) ZDT2 (b) ZDT3

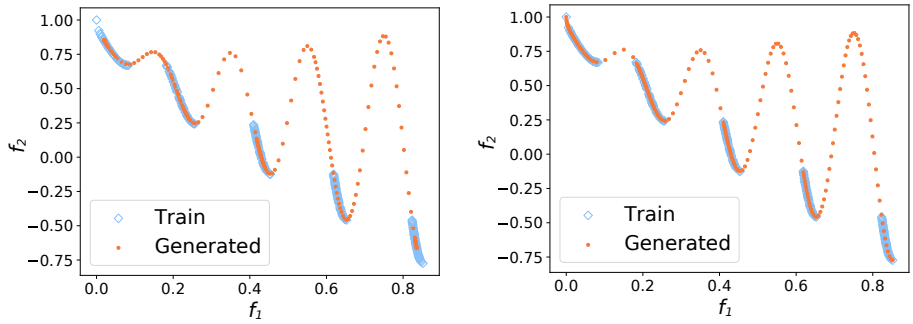
Fig. 4: GPR approach on continuous gap in middle of PO front for ZDT2 and ZDT3 problems, demonstrating Task 2.

a uniform sample of 200 uniformly distributed set of pseudo-weight vectors and apply our trained models using EMO-obtained PO points to find respective **x**-vectors, compute their **f**-vectors, and then plot them in Figure 5. It is clear that for pseudo-weights resulting a PO solution, our trained models are able to find them, but for pseudo-weights in natural gaps, trained models have produced dominated solutions, confirming that gaps observed in EMO solutions are true gaps and no PO solution exists there. Thus, our proposed approach can also be used to confirm reality of gaps in EMO-obtained PO fronts.

Constrained two-objective problems: Tables 1 and 2 show that the GPR approach is able to produce PO points from pseudo-weights with smaller error compared to DNN as well. To demonstrate, we present random and gap point discovery tasks (Tasks 1 and 2, respectively) for BNH and OSY problems in Figure 6.

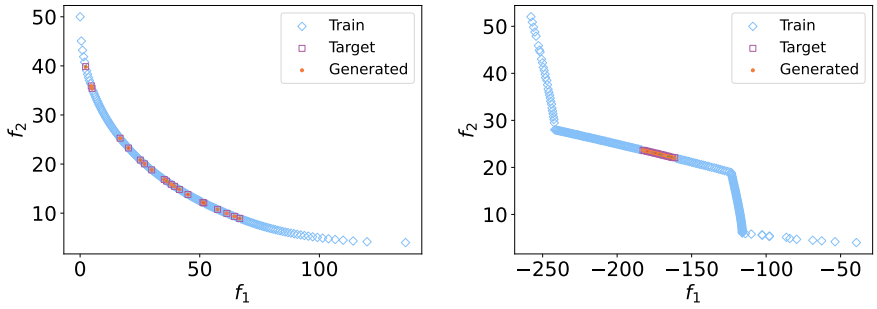
4.2 Three-objective problems

Owing to better results with GPR approach for two-objective problems, we use only GPR for three and many objective problems. Table 3 shows **x** and **f** errors along with their standard deviations.



(a) DNN on ZDT3 (b) GPR on ZDT3

Fig. 5: Pseudo-weights on true gaps produce dominated solutions, but pseudo-weights on true PO front produce PO solutions, demonstrating Task 3.



(a) Random points on BNH (b) Continuous gap points on OSY

Fig. 6: GPR results showing discovery of PO points by the GPR approach on two constrained problems, demonstrating Tasks 1 and 2.

Figure 7 shows random and continuous gap predictions for DTLZ2 problem. We see that the ML model is able to reasonably predict **x**-vectors (hence, **f**-vectors) for unseen pseudo-weights with low error.

Constrained three-objective problems: From Table 4, we see that prediction of PO solutions on constrained problems (carside impact and crashworthiness) can be achieved by the GPR approach with low errors in **x** and **f** space with low standard deviations.

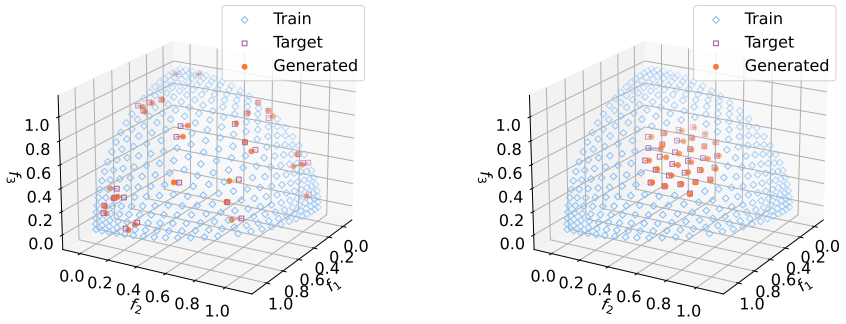
Validating Task 4: Next, we consider a simulated case in which an EMO/EMaO produces a set of PO solutions which are not uniformly distributed. In this case as well, we can train a GPR model and supply pseudo-weights at the part with low-density of solutions and expect our model to predict PO points there. Figure 8 shows that the GPR approach can fill in additional points in such low-density regions for DTLZ2 and crashworthiness problems.

4.3 Many-objective Optimization Problems

Finally, we apply our GPR approach to two five and 10-objective DTLZ2 and C2-DTLZ2 problems to demonstrate proof-of-principle results on many-objective

Table 3: Performance of GPR approach on three-objective problems.

Problem	M	Gap Type	Mean $\mathbf{x}$	Mean $\mathbf{f}$	Std. Dev. in $\mathbf{x}$	Std. Dev. in $\mathbf{f}$
DTLZ2	3	Continuous	6.981E-03	3.126E-02	2.449E-03	9.535E-03
		Edge	1.022E-02	5.968E-02	3.431E-03	1.962E-02
		Random	3.419E-03	1.153E-02	2.494E-03	7.168E-03
		Sparse	3.917E-03	2.238E-02	1.594E-03	8.891E-03
WFG2	3	Continuous	5.600E-02	6.514E-02	1.274E-01	1.703E-01
		Random	4.773E-02	4.922E-02	1.220E-01	1.732E-01
Carside	3	Random	1.171E-02	3.957E-03	9.080E-03	1.471E-02
Crashworthiness	3	Sparse	1.009E-02	7.785E-03	1.388E-02	3.903E-02



(a) Random points (Task 1) (b) Continuous gap points (Task 2)

Fig. 7: Random and continuous gap points by the GPR approach on three-objective DTLZ2 problem.

optimization problems. Table 4 shows small error and standard deviation of error by the GPR approach.

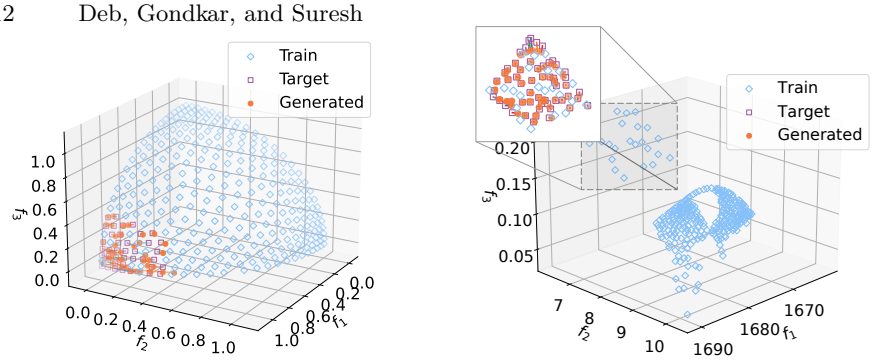
Figure 9 shows the parallel coordinate plot (PCP) on these two problems with a few randomly chosen pseudo-weight vectors to demonstrate that the obtained $\mathbf{x}$-vectors (and their $\mathbf{f}$-vectors) produce near PO solutions.

5 Conclusions

We have shown that machine-learning (ML) models can be used to learn patterns between pseudo-weights and corresponding $\mathbf{x}$-vectors and generate new points on the Pareto front without doing an additional optimization. We have demonstrated that this method is scalable to many-objective test and real-world problems. This proof-of-concept study paves the way for using the proposed method as part of an EMO task for generating a better distributed ND fronts. Owing to the fact that the ML models are conditioned on pseudo-weights, the

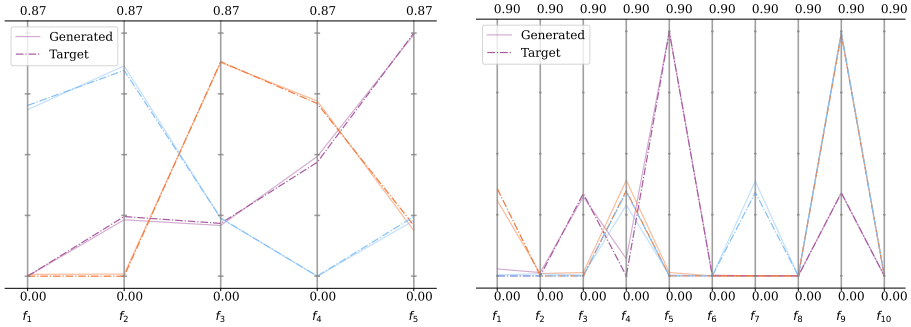
Table 4: Performance of GPR approach on many-objective problems.

Problem	M	Test Data	Mean $\mathbf{x}$	Mean $\mathbf{f}$	Std. Dev in $\mathbf{x}$	Std. Dev in $\mathbf{f}$
DTLZ2	5	Random	9.887E-03	1.382E-02	9.155E-03	8.668E-03
DTLZ2	10	Random	4.582E-02	1.418E-02	3.716E-02	8.500E-03
C2DTLZ2	5	Random	1.020E-02	1.323E-02	1.028E-02	7.448E-03
C2DTLZ2	10	Random	4.532E-02	1.418E-02	3.552E-02	9.490E-03



(a) DTLZ2 (b) Crashworthiness

Fig. 8: Additional solutions are supplied by the GPR approach for two three-objective problems at region of low-density of solutions, demonstrating Task 4. A few blue points were found by EMO on a part of PO front, but our ML approach has nicely replenished them.



(a) 5-objective DTLZ2 (b) 10-objective C2DTLZ2

Fig. 9: PCP plots showing true PO and GPR-trained solutions are close for a few pseudo-weight vectors for 5-obj DTLZ2 and 10-obj. DTLZ2 problems.

proposed method can be readily used for decision-making by the user without the need for further optimization.

The current study can be extended as a comparison with optimization based gap-filling methods, like reference direction based EMO algorithms. Applications to more complex and real-world problems will fully evaluate the potential of the proposed approach. Also, the modeling approaches can be improved to satisfy learning of constraints and variable bounds during the training process. For example, specific activation functions (such as, ReLU) can be used for the output layer of DNNs to restrict outputs to variable bounds. While constraint satisfaction was enforced in our results here, it would be a challenging task to include constraint satisfaction in the training process, since all PO solutions (training data) are expected to be feasible. Nevertheless, this proof-of-principle study opens up a unique use of machine learning methods in assisting multi-objective optimization.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. p. 2623–2631. KDD '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3292500.3330701, https://doi.org/10.1145/3292500.3330701
2. Coello, C.A.C., VanVeldhuizen, D.A., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Boston, MA: Kluwer (2002)
3. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part I: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation **18**(4), 577–601 (2014)
4. Deb, K., Srinivasan, A.: Innovization: Innovating design principles through optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006). pp. 1629–1636. New York: ACM (2006)
5. Deb, K.: Multi-objective optimization using evolutionary algorithms (2001)
6. Deb, K., Roy, P.C., Hussein, R.: Surrogate modeling approaches for multiobjective optimization: Methods, taxonomy, and results. Mathematical and Computational Applications **26**(1) (2021). https://doi.org/10.3390/mca26010005, https://www.mdpi.com/2297-8747/26/1/5
7. Deb, K., Sundar, J.: Reference point based multi-objective optimization using evolutionary algorithms. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. pp. 635–642 (2006)
8. Farias, L.R.C., Araújo, A.F.R.: IM-MOEA/D: An inverse modeling multi-objective evolutionary algorithm based on decomposition. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 462–467 (2021). https://doi.org/10.1109/SMC52423.2021.9658650
9. He, C., Huang, S., Cheng, R., Tan, K.C., Jin, Y.: Evolutionary multiobjective optimization driven by generative adversarial networks (GANs). IEEE Transactions on Cybernetics **51**(6), 3129–3142 (2020)
10. Jain, H., Deb, K.: 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 (2014)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
12. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: Enhanced innovized progress operator for evolutionary multi- and many-objective optimization. IEEE Transactions on Evolutionary Computation **26**(5), 961–975 (2022). https://doi.org/10.1109/TEVC.2021.3131952
13. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32. pp. 8024–8035. Curran Associates, Inc. (2019)

14. Pellicer, P.V., Escudero, M.I., Alzueta, S.F., Deb, K.: Gap finding and validation in evolutionary multi-and many-objective optimization. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. pp. 578–586 (2020)
15. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. Evolutionary Computation, IEEE Transactions on **11**(6), 712–731 (2007)