Complex & Intelligent Systems (2022) 8:5287–5308
https://doi.org/10.1007/s40747-022-00759-w

ORIGINAL ARTICLE

Machine learning-based framework to cover optimal Pareto-front in many-objective optimization

Azam Asilian Bidgoli[1] · Shahryar Rahnamayan[1] · Bilgehan Erdem[1] · Zekiye Erdem[1] · Amin Ibrahim[1] · Kalyanmoy Deb[2] · Ali Grami[1,3]

Received: 17 January 2022 / Accepted: 15 April 2022 / Published online: 16 May 2022

Abstract

One of the crucial challenges of solving many-objective optimization problems is uniformly well covering of the Pareto-front (PF). However, many the state-of-the-art optimization algorithms are capable of approximating the shape of many-objective PF by generating a limited number of non-dominated solutions. The exponential increase of the population size is an inefficient strategy that increases the computational complexity of the algorithm dramatically—especially when solving many-objective problems. In this paper, we introduce a machine learning-based framework to cover sparse PF surface which is initially generated by many-objective optimization algorithms; either by classical or meta-heuristic methods. The proposed method, called many-objective reverse mapping (MORM), is based on constructing a learning model on the initial PF set as the training data to reversely map the objective values to corresponding decision variables. Using the trained model, a set of candidate solutions can be generated by a variety of inexpensive generative techniques such as Opposition-based Learning and Latin Hypercube Sampling in both objective and decision spaces. Iteratively generated non-dominated candidate solutions cover the initial PF efficiently with no further need to utilize any optimization algorithm. We validate the proposed framework using a set of well-known many-objective optimization benchmarks and two well-known real-world problems. The coverage of PF is illustrated and numerically compared with the state-of-the-art many-objective algorithms. The statistical tests conducted on comparison measures such as HV, IGD, and the contribution ratio on the built PF reveal that the proposed collaborative framework surpasses the competitors on most of the problems. In addition, MORM covers the PF effectively compared to other methods even with the aid of large population size.

Keywords Optimization · Many-objective · Pareto-front · Reverse mapping · Machine learning

Abbreviations

ML	Machine learning
MORM	Multi-objective reverse mapping
EA	Evolutionary algorithms
MOOP	Many-objective optimization problem
NSGA-III	Non-dominated sorting genetic algorithm (version III)
IM-MOEA	Inverse modeling-based multi-objective evolutionary algorithm
MOEA/D	Decomposition-based multi-objective evolutionary algorithm
PF	Pareto-front
OBL	Opposition-based learning
LHS	Latin hypercube sampling
ANN	Artificial neural network
MSE	Mean square error
NDS	Non-dominated sorting

Variables

$\boldsymbol{x}$	A vector of decision variables
$\boldsymbol{y}$	A vector of objective values
X	A set of decision vector

✉ Azam Asilian Bidgoli
azam.asilianbidgoli@ontariotechu.ca

[1] Department of Electrical, Computer, and Software Engineering, Ontario Tech University, Oshawa, Canada

[2] Computational Optimization and Innovation (COIN) Laboratory Departments of Electrical and Computer Engineering, Computer Science and Engineering, and Mechanical Engineering, Michigan State University, East Lansing, USA

[3] Department of Electrical, Computer, and Software Engineering, Ontario Tech University, Oshawa, Canada

Y	A set of objective vector
x_i	A decision variable
y_i	An objective value
F	Set of Objective functions
f_i	An objective function
D	Number of variables
M	Number of objectives
$\check{y}$	Opposite of y
y^{po}	Partial opposite of y
M	A trained model

Introduction

Evolutionary algorithms (EAs) have been very powerful and well established approaches to solve many-objective optimization problems (MOOPs). However, when the number of objectives are high, they struggle to find well-covered and well-distributed solutions. Moreover, it has been shown that more than 90% of the randomly generated initial population are non-dominated when the number of objectives are many [19]. Furthermore, the small size of a population is not able to cover the large-scale hyper PF surface and as a result, this causes a sparse PF set. One way to reduce the impact of sparsity in many-objective optimization is to increase the initial population size. However, when the population size is increased, the optimization process required to obtain converged and well-distributed PF is computationally expensive [31]. Moreover, the theoretical results presented by Chen et al. have revealed that large population, depending on a problem characteristics, may not always be useful and even it can degrade the performance of EAs [5]. Furthermore, it would be challenging to tackle the sparsity issue as the dimension of the problem gets large since large sparse regions occur in the PF with the increasing number of dimensions. In addition, since optimization algorithm has difficulties when the PF has sparse regions and consequently decision-maker cannot reach all the points in the region of the interest (ROI) of PF.

Recently, Deb and Srinivasan [13] introduced the idea of innovization in multi-objective optimization to discover the knowledge and patterns hidden in the PF. Although their study focused on the discovery of important design principles related to decision variables and objectives after the optimization process is done. Innovization can be used as part of the optimization process to improve the quality of the Pareto-optimal solutions [12,30]. Subsequent to the introduction of innovization, there were several studies focusing on the idea of incorporating of learning algorithms during the optimization process to gain information that leads to a set of improved solutions [4,8,27].

This study proposes a novel framework to fill sparse regions in many-objective optimization by employing machine learning (ML) after the optimization process is complete. The proposed framework uses a machine learning algorithm to approximate the mapping between the PF solutions and decision variables. This information is then used to generate new candidate solutions without the need for re-running optimization algorithm. As a result, the sparsity issue of PF can be addressed by the proposed collaborative ML and MOO framework.

There is a growing interest in using the machine learning techniques to enhance the performance of optimization algorithms. Gaspar et al. presented a hybrid multi-objective EA to accelerate the search using artificial neural networks (ANN) [17]. The aim of their study was to reduce the number of fitness evaluation by approximating global fitness functions; hence machine learning is used to make a local search to find better individuals. Adra et al. used the same framework for optimizing the aircraft control system design [1]. Another similar study proposed an algorithm to speed-up the multi-objective genetic algorithm with constraints using ANN [18]. The authors introduced an approximated function of the original objective function using ANN, and used this function during the optimization to reduce the computational time of the evaluations. Kobayashi et al. presents a new mechanism which helps to maintain the diversity of solutions using ANN [23]. A set of NSGA-III solutions are used to train the ANN to find an approximation function as a surrogate model. The study used this approximated function to relocate the population uniformly. The use of machine learning to construct a proxy model for many-objective optimization problems is presented in several studies [3,26,33,47]. A proxy model can be created on simulation results of an experimental problem to be then solved by a multi-objective optimization algorithm and consequently to reduce the number of lab-based expensive experiments. Alternatively, for an expensive fitness function, a proxy model is built to approximate the value of objectives rather than evaluate them directly. Generally speaking, most of the studies in the literature have used the machine learning to improve the evolution process by surrogate model and they are not targeting adding new non-dominated solutions to the PF. Surrogate models construct a function mapping from the decision space to objective space. They estimate the fitness value of candidate solutions. Therefore, they are helpful when no explicit fitness function exists or when the original fitness function is computationally very expensive. On the contrary, the reverse mapping algorithm constructs a mapping model from the objective space to decision space. In [6], a reverse model algorithm, IM-MOEA, is introduced based on the Gaussian process to estimate the objective values of some random samples from objective space. They create Gaussian model for each objective and each cluster of population. Therefore, the computational complexity increases exponentially. In addition, they use descriptive models to estimate the distribution of

the points and for accomplishment, they have been forced to consider some assumptions such as the independency among the objectives.

One of the major difficulties of large-scale many-objective optimization is to obtain acceptable solutions with a reasonable size of the population. Therefore, it is necessary to reduce the number of evaluations to reach this goal [16]. This difficulty might be alleviated using a surrogate model with machine learning methods. The idea of this study is highly influenced by the innovization idea and reverse models because of seeking possible ways to gain information from the PF solutions. In this paper, we proposed a reverse mapping from objective space to decision space. To this end, a learning model is trained to map the solutions on the PF, resulted by an optimization algorithm, to decision variables. Using this model, the decision variables can be estimated for the desired values of objectives. Therefore, a cloud of non-dominated solutions can be generated using a variety of techniques while their decision values can be obtained by the trained model. Two categories of generative techniques applied on the decision or objective spaces can create new points with the help of trained reverse mapping model and problem's objectives. As a result, decision-maker can select an acceptable solution from a comprehensive set of PF solutions by considering trade-off among own desired objective values.

The main contributions of the proposed framework to tackle the mentioned problems in the population-based algorithms are

1. A novel reverse model is proposed to map the objective space to decision space.
   - Capability of comprehensively covering the Optimal PF with no need to rerun the optimizer.
   - with low cost, there is the capability of increasing PF solutions to the desired number (i.e., as many as possible).
   - Enhancement of the uniform distribution of PF solutions which provides the decision-makers with a higher resolution of trade-off solutions with a low sparsity over PF regions.
   - It can be utilized with any MOO algorithm (i.e., classical or metaheuristic one) which can offer an approximate PF set.
   - There is a capability of getting a preferred solution(s) (i.e., desired objective values) from the user to find corresponding solution(s). Thus, it can help decision-makers to selectively fill a sparse Region-of-Interest (RoI) interactively by conducting a well-defined controlled procedure with no need to rerun the optimizer.
2. Instead of random selection of points in objective space, several sampling techniques are proposed for inverse sampling which leads to generating more non-dominated solutions to reduce the level of sparsity on surface of hyper curve of PF.
3. A reference-line based scheme is designed to select a set of well-distributed solutions after finishing of optimization.
4. Strongly supporting scalability on the number of objectives; the proposed model performs very well with many objective cases, because objective values are the inputs to the introduced reverse mapping model; dislike other models which are faced with the difficulty when the number of objectives increases.
5. Systematic experiments conducted to compare the proposed algorithm with the-state-of-the-art MOEAs on ten MOPs are described.

Finally, we want to remark that in most cases, real-life problems have many objectives and even more decision variables. This leads to the higher dimensionality problem in decision and objective spaces. Most of the similar studies try to avoid complexity, however, we take advantage of two types of complexity in our approach; (i) higher dimensions in objective space, (ii) approximation errors in reverse mapping. First, we use the objective function values as inputs of the machine-learning model, therefore having more objectives leads to accurate learning (i.e., in opposite preference of other methods). Second, approximation error acts in our benefit since it helps as a local search in finding new feasible candidate decision variables and solutions.

The remainder of the paper is organized as follows. The next section provides the background review of the study. The technical description of the proposed framework is presented in the subsequent section followed by which the experimental setup and experimental results of the proposed framework on well-known multi-objective benchmark test problems with 3 and 5 objectives are presented. In addition to the benchmarks, the performance of proposed method on two well-known real-word problems is investigated in the penultimate section. Finally, the concluding remarks are provided.

Literature review

Generally speaking, evolutionary-based multi-objective optimization algorithms can be divided into three categories: dominance-based, decomposition-based, and indicator-based methods [41]. Dominance-based methods attempt to find the solutions that optimize the objective functions using the dominance concept. One of the stat-of-the-art algorithms from this category is NSGA-IIII [10]. According to this method, the candidate solutions in population are selected based on the reference points distributed in the objective space. On the contrary, the decomposition-based methods

decompose the whole search space into smaller subproblems and solve all of them simultaneously. Therefore, the convergence rate of the algorithm is significantly improved, which enhances the diversity of the obtained solutions. MOEA/D [48] is one of the well-known algorithms in this category. The indicator-based methods evaluate the fitness of each solution by assessing an indicator (such as hypervolume) to improve the convergence and diversity criteria simultaneously. All traditional multi-objective algorithms belong to aforementioned categories focus on the proposing a selection strategy when attempting to adopt single-objective EAs to solving MOEAs. Whereas designing effective reproduction strategies that explicitly focus on covering PF by generating well-distributed solutions has not been paid attention properly.

Apart from this categorization, the traditional multi-objective optimization algorithms use the reproduction strategies to generate new individuals in original decision space. To this effect, there are a variety of methods which can significantly improve the performance of existing algorithms by designing new generative operators [20,40,46]. In contrary, some MOEAs generate offsprings using probabilistic models to characterize the promising solutions and approximate the Pareto-front instead of utilizing generative operators. In fact, these methods use the knowledge from the approximated Pareto-front to guide the MOEAs. Reverse models belong to these category which construct a model to map the objective space to decision space. This idea provides capability for MOEAs to generate desired candidate solutions in objective space explicitly.

IM-MOEA [6] constructs a model using Gaussian process to map the objective space to the decision space. Consequently, the authors generate candidate solutions in the objective space. However, they require $K \times M$ modes where K is the number of population partitions and M is the number of objective. This is thus an obstacle against scalability in term of number of objectives. This algorithm suffers low accuracy and difficulty in dealing with MOPs with irregular PFs because it is not able to generate distribution in sparse regions. In [44], a method is proposed based on the idea of IM-MOEA but the reference vectors are generated adaptively and additionally nonrandom grouping strategy are employed. In [25], a model based on an incremental Gaussian mixture model guides the search procedure. They fed the learning model with all new solutions generated during the evolution to adaptively discover the structure of the Pareto-optimal set. In addition, to conduct the Sensor Ontology Matching (SOM) process to find the mappings among diverse sensor data, an Improved IM-MOEA (I-IM-MOEA)-based matching technique is proposed [44]. An adjusted selection mechanism is employed to tackle the problems on irregular PFs such as reduction in PF solutions. Moreover, a dynamic Reference Vectors is proposed to decrease the computational resources and improve the efficiency of the algorithm. The potential of reverse models for solving dynamic multi-objective optimization problems is investigated in [49]. An inverse-based Gaussian process maps the historical optimal solutions from the objective space to the decision space.

A decomposition-based reverse model algorithm which uses k-means for grouping in the objective space is introduced in [15]. Furthermore, a selection criterion based on decomposition that selects the most appropriate reference vectors is proposed. All reviewed studies attempt to find a distribution on PF which have to make some assumptions such as independency among the objective functions while our proposed method alleviates these drawbacks as it generates a model using the learning process. Moreover, these method are dependent on the shape of PF and are not usually able to generate distribution model on sparse are of PF. Whereas our proposed method can train a model using a limited number of points.

Background review

In this section, the concepts related to the study have been explained. In addition to the definition of many-objective optimization and ANN, the explanation of three techniques which are utilized to generate new points in both decision and objective spaces is provided.

Many-objective optimization

Many-objective optimization targets handling more than three conflicting objectives. Multi-objective optimization algorithms have been greatly expanded to tackle many-objective problems. The use of EAs has been very promising for solving such problems. The population-based nature of these algorithms results in generating a set of candidate solutions at each run of the algorithm. Collaboration of individuals to make an optimal Pareto-front is the core reason of success of population-based algorithms compared to single-solution algorithms.

Definition 1 Many-objective optimization [2]

$$\begin{aligned} &\text{Min/Max } F(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})] \\ &\text{s.t.} \quad L_i \leq x_i \leq U_i, i = 1, 2, \ldots, d. \end{aligned} \tag{1}$$

Subject to the following equality and/or inequality constraints.

$$\begin{aligned} &g_j(\boldsymbol{x}) \leq 0 \quad j = 1, 2, \ldots, J \\ &h_k(\boldsymbol{x}) = 0 \quad k = 1, 2, \ldots, K, \end{aligned} \tag{2}$$

where M is the number of objectives, d is the number of decision variables (i.e., dimension), and the value of each

variable, $\boldsymbol{x}_i$, is in interval $[L_i, U_i]$ (i.e., box-constraints). f_i represents the objective function, which should be minimized or maximized.

Due to the conflicting of objective functions in a multi- or many-objective optimization problems, the definition of the optimality is not as simple as the single-objective case. Therefore, it is required to make a trade-off decision among objective functions. One of the commonly used concepts for comparing candidate solutions in such problems is dominance.

Definition 2 Dominance Concept If $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ and $\acute{\boldsymbol{x}} = (\acute{x}_1, \acute{x}_2, \ldots, \acute{x}_d)$ are two vectors in a minimization problem search space, $\boldsymbol{x}$ dominates $\acute{\boldsymbol{x}}$ ($\boldsymbol{x} \succ \acute{\boldsymbol{x}}$) if and only if

$$\begin{aligned} &\forall i \in \{1, 2, \ldots, M\},\ f_i(\boldsymbol{x}) \leq f_i(\acute{\boldsymbol{x}}) \wedge \\ &\exists j \in \{1, 2, \ldots, M\} : f_j(\boldsymbol{x}) < f_j(\acute{\boldsymbol{x}}) \end{aligned} \tag{3}$$

This concept defines the optimality of a solution in a multi-objective space. Candidate solution $\boldsymbol{x}$ is better than $\acute{\boldsymbol{x}}$ if it is not worse than $\acute{\boldsymbol{x}}$ in any of the objectives and at least it has a better value in one of the objectives. All solutions that are not dominated using any other solution called non-dominated solutions; they create the PF set. Many-objective algorithms attempt to find these solutions by utilizing generating strategies/operators and selection schemes. The non-dominated sorting (NDS) algorithm [11] is one of the popular selection strategies which works based on the dominance concept. It ranks the solutions of the population in different levels of optimality, called Pareto. The algorithm starts with determining all non-dominated solutions in the first rank. To identify the second rank of individuals, the non-dominated vectors are removed from the set to process the remaining candidate solutions in the same way. Non-dominated solutions of this step make the second level of individuals (second Pareto). Thereafter, the second ranked individuals will be removed to identify the third Pareto. This process will continue until the whole individuals are grouped into different levels of Pareto.

Artificial neural network

Artificial Neural Network (ANN) is one of the most powerful tools for pattern recognition and data mining that is inspired by human brain [45]. Nowadays, there has been growing interest in using ANN in a variety of applications for developing a learning model [32,37,50]. The network structures are made up of two components: neurons and weighted connections among neurons. The feed-forward network is a popular category of ANNs with an input layer for feeding input data, an output layer for specifying the output of classification or regression, and one or more hidden layers between input and output layers for the learning process. The learning process in the network is based on finding the best connection weights and thresholds of neurons for hidden and output layer with the goal of achieving minimum error for predicting the output of test data. To train a network, training data is fed to the ANN. The error can be computed using a desired metric such as Mean Square Error (MSE) defined as follows.

$$\text{MSE} = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2, \tag{4}$$

where n is the number of samples, y_i is the desired Neural Network output, and $\hat{y}_i$ is the neural network output. The weight updates are performed using an optimization method such as Gradient Descent. The best values for the weight connections and threshold are calculated to minimize the corresponding error, called loss value. This process continues until the algorithm meets a predefined criterion such as reaching to a specific number of iterations or desired error value.

Opposition-based learning (OBL)

Opposition-based learning (OBL) is a technique to solve the optimization/learning problems effectively and efficiently [34]. Literature shows that OBL augments the diversity of generated solutions for an optimization problem by exploring more regions of the search space in both current and opposite directions. In this study, OBL is used as a parallel approach to produce more and diverse points on the PF cloud. In the following, three types of the opposition-based operators are briefly explained.

Definition 3 Type I opposition Let $\boldsymbol{x} = (x_1, \ldots, x_D)$ be a point in D-dimensional space and $x_i \in [a_i, b_i], i = 1, 2, \ldots, D$. The opposite of $\boldsymbol{x}$ is defined by $\breve{\boldsymbol{x}} = (\breve{x}_1, \ldots, \breve{x}_D)$ as follows: [28]:

$$\breve{x}_i = a_i + b_i - x_i. \tag{5}$$

Definition 4 Type II opposition For the point $\boldsymbol{x} = (x_1, \ldots, x_D)$ we have $\boldsymbol{y} = f(\boldsymbol{x}) \in [Y_{\min}, Y_{\max}]$, the min-max opposition computation for Type-II opposition is defined as [36]:

$$\breve{\boldsymbol{y}} = Y_{\min} + Y_{\max} - \boldsymbol{y}. \tag{6}$$

Definition 5 Partial opposition In a multi-dimensional space, the partial opposite point $\boldsymbol{y}^{\text{po}} = (y_1, \breve{y}_2, y_3, \ldots, \breve{y}_D)$ is generated by opposing only a portion of dimensions [29] which can be selected randomly. So, we can define the partial opposite set of points $\boldsymbol{y}^{\text{po}}$ as follows:

$$\boldsymbol{y}^{\text{po}} = \boldsymbol{y}_1^{\text{po}}, \boldsymbol{y}_2^{\text{po}}, \ldots, \boldsymbol{y}_D^{\text{po}} = \begin{Bmatrix} y_1 & \breve{y}_2 & \breve{y}_3 & \cdots & \breve{y}_D \\ \breve{y}_1 & \breve{y}_2 & y_3 & \cdots & y_D \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \breve{y}_1 & y_2 & \breve{y}_3 & \cdots & \breve{y}_D, \end{Bmatrix} \tag{7}$$

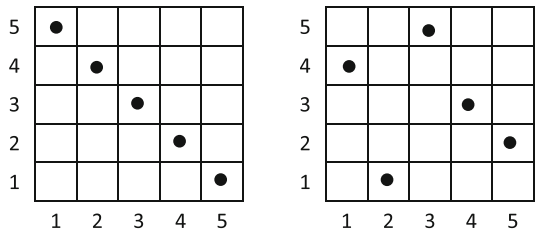


Fig. 1 Two sample LHS configurations with two variables in five intervals

where $\check{y}_i$ is the opposite value of y_i. Obviously, there are 2^D partial opposite combinations for a D-dimensional vector.

Latin hypercube sampling

Latin Hypercube Sampling (LHS) is a stratified sampling technique to generate near-random samples from a probability distribution [38]. To generate N samples from D variables, the range of each variable is divided into N equally probable intervals. From each interval, a random sample is selected to cover a large space of space. Since each sample of each variable can be paired in a random manner with other variables, for given values of N and D, there exist $(N!)^{D-1}$ possible interval combinations. Two examples of different combinations are illustrated in Fig. 1 representing a LHS in two dimensions with five intervals.

Proposed algorithm: MORM

The main goal of the proposed method is to comprehensively cover the PF using a trained model on a resulted small set of solutions from an optimization process. In many-objective optimization problems, obtaining well-covered PF is a challenging task especially when the number of objectives increases and PF is a hyper-surface as a result. The main key idea is to train a model on optimal PF set obtained from an optimization process to map the decision variables of the resulted objective values. In this way, generating numerous points in objective space leads to the capability of trained model to estimate the corresponding decision variables. Therefore, a cloud of non-dominated solutions can be obtained with no need to rerun an optimization process. For this purpose, a sparse PF is initially achieved by an arbitrary many-objective optimization algorithm, either by a classical or meta-heuristics method. Considering a learning algorithm, i.e., regression technique, the inputs of the model are the objective values on provided PF and its outputs are corresponding decision variables. The learning process aims to find a mapping model between objectives and variables values to predict the decision vector for a given objective vector. Accordingly, by providing more points around the resulted PF, we are able to estimate the decision values for these objective values. In addition, the constructed model gets improved over iterations, because the training points increase gradually. This approach is called many-objective reverse mapping (MORM) as constructed model conducts a mapping from objective space to decision space in reverse direction of an objective function which is a mapping from the decision to objective space.

The trained model is utilized to predict the decision values for those points generated close to resulted PF from optimization task. Afterward, the true objective values for predicted decision variables can be achieved by objective function in a direct mapping. Newly generated pairs of decisions and their corresponding objective values are utilized to improve the trained model. This procedure is the fundamental part of the proposed framework. The process is repeated to generate as much as the required non-dominated solutions. To generate the approximate points in both decision and objective spaces, several techniques including OBL and LHS are employed. The details of main steps of the proposed framework are provided as follows.

1. Optimization. At the first step of the framework, a many-objective optimization algorithm is utilized to generate the initial PF for the given problem. The resulted non-dominated solutions are exploited for two purposes. Firstly, a machine learning model is trained on objective values to predict the decision values of the following generated points. In addition, these points can be input for other auxiliary techniques such as OBL and LHS to generate more points in objective and decision space. Hence, regardless of the type of the optimization algorithm, any method which is able to obtain an initial optimal PF of the problem can be employed. Therefore, the output of this step is a PF (i.e., a set of decision variables (X) and objective values (Y) resulted from a conventional many-objective optimization algorithm.

$$\begin{aligned} \text{PF} : (X, Y) = \{(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x} = (x_1, x_2, \ldots, x_D), \\ y = (y_1, y_2, \ldots, y_M) \quad \text{and } \boldsymbol{y} = F(\boldsymbol{x})\}, \end{aligned} \tag{8}$$

where $F(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})]$ is a set of objective functions.

2. Reverse mapping. In this phase, a machine learning algorithm is used for the surrogate model to conduct the reverse mapping from objective space to decision space. Obviously, the fitness function maps the variables of decision space to their objective values. Conversely, in the reverse direction, a learning model, i.e., regression model, can be trained to map the objectives into decision variables. To this end, the generated PF using an optimization algorithm in the previous step is provided as training data for the utilized learning algorithm. The objective values of non-dominated solutions are inputs of the model and the decision variables

form the outputs. After being given a sufficient number of samples, the model becomes capable of predicting decision variables from associated objective values. The estimated decision values of the trained model are evaluated by the fitness function to get the true value of objectives. This step leads to generating a set of points different from the training data because similar to every machine learning algorithm, an approximation model yields the approximation error; this situation results in a cloud of non-dominated solutions. After completing the training process, the generated objective values are entered into the constructed model as input data to get approximate decision variables. The process of mapping between decision and objective spaces is presented in Fig. 2. Therefore, the output of this step is a trained model (M) on the resultant PF from the previous step which maps the objective values ($\boldsymbol{y}$)into decision variables ($\boldsymbol{x}$) as follows.

$$\boldsymbol{x} = \boldsymbol{M}(\boldsymbol{y}, \beta) : \boldsymbol{y} \rightarrow \boldsymbol{x}, \tag{9}$$

where β is the a set of model parameters. It is obvious that if we feed the model using the current objective values, Y, a set of new decision variables can be obtained so that $X_M = \boldsymbol{M}(Y, \beta)$. Correspondingly, the true objective values of X_M can be obtained using fitness evaluation, $Y_M = F(X_M)$.

3. Generative techniques. To produce a cloud of non-dominated solutions for an optimization problem, other auxiliary methods can be utilized along with the optimization process. These techniques generate new points close to discovered solutions on PF. Moreover, new points can also be created in the decision space. Once the new points are generated in the objective space, the trained model is employed to map the objective values to decision variables reversely. For this purpose, two techniques are employed including OBL and LHS. The details of operating these techniques are provided in the following.

OBL As mentioned in the background review, there are two types of OBL, namely Type I and Type II which are utilized in the following ways.

Type I OBL can be applied in decision space. In this case, the opposite of decision variables of non-dominated solutions, X_{opI}, are computed to discover more candidate solutions with higher diversity. In a direct mapping, for the newly discovered decision vectors, the objective values are computed using fitness functions so that $Y_{\text{opI}} = F(X_{\text{opI}})$. As a result, a set of candidate solutions are produced using this technique. OBL leads to exploring the search space in both current and opposite directions to enhance the diversity and coverage of the PF. On the other hand, type II OBL and its variant, partial type II OBL, produces a set of new points in objective space, Y'_{opII} and Y'_{popII}, respectively. These points are in the opposite direction of objective vectors of the resulted PF.

Since these points are generated in objective space, the corresponding decision variables, x_{opII} and x_{opII}, should be estimated as follows:

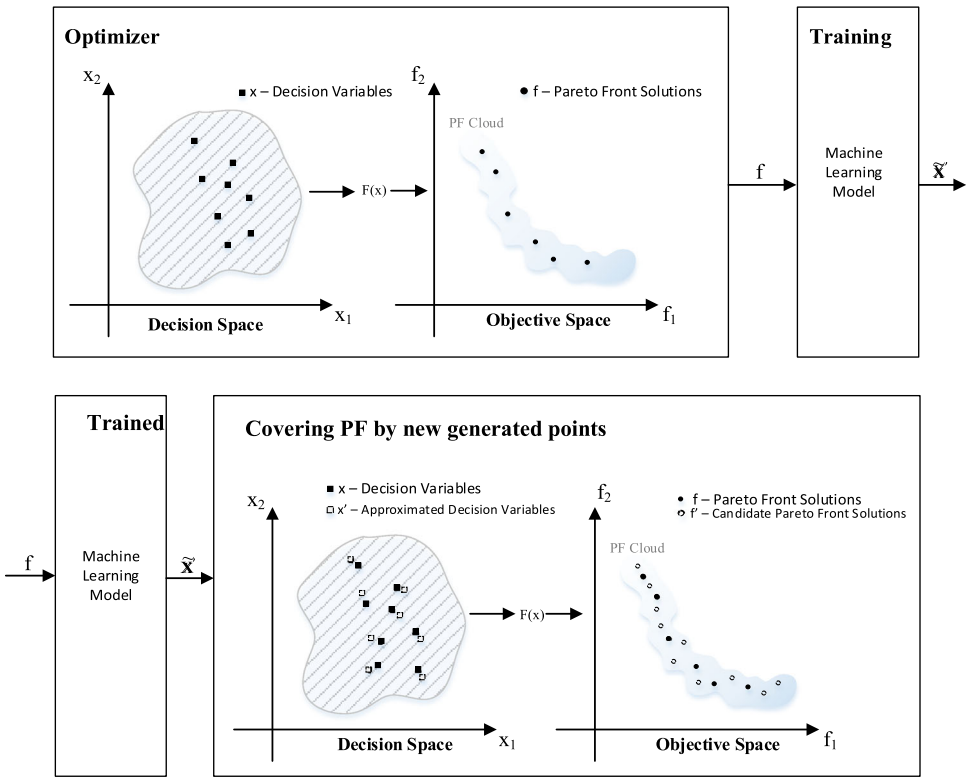


Fig. 2 Mapping between decision and objective spaces. In the upper subfigure, the optimizer estimates an initial PF, and the fitness function maps the decision space to objective space. The Machine learning model is constructed on PF which is the output of the optimizer. In the lower subfigure, the approximate decision variables are the output of the trained model and new candidate solutions are added then non-dominated ones have been remained as the improved PF

$$X_{\text{opII}} = \boldsymbol{M}(Y'_{\text{opII}}, \beta) \quad and \quad X_{\text{popII}} = \boldsymbol{M}(Y'_{\text{popII}}). \tag{10}$$

To compute the true objective values of approximate decision vectors, fitness evaluation is utilized and this is in fact the only time we need to call the fitness function so that $Y_{\text{opII}} = F(X_{\text{opII}})$ and $Y_{\text{popII}} = F(X_{\text{popII}})$. By a well-trained mapping model, the new objective vectors are expected to be close to the opposite of the original non-dominated solutions. Consequently, a set of decision and objective vectors are resulted using these techniques.

LHS It is another sampling technique to generate a pool of new samples in objective space. Using the aforementioned procedure, a set of objective vectors are sampled from sub-regions of PF intervals, Y'_{LHS}. The sample spaces are resulted from splitting the interval of each objective of non-dominated solutions. This technique attempts to distribute samples evenly over the sample space. The set of random numbers generated by the LHS method are the appropriate representatives of the real variability rather than traditional random sampling. Hence, it leads to better exploration of PF and generation of new non-dominated points with a higher probability. Similar to other generative methods applied in objective space, the reverse mapping mode takes the resulted objective vector as input to estimate the corresponding decision vectors, X_{LHS}, as follows:

$$X_{\text{LHS}} = \boldsymbol{M}(Y'_{\text{LHS}}, \beta). \tag{11}$$

Then, the valid fitness values of predicted decision vector are computed using the fitness function so that $Y_{\text{LHS}} = F(X_{\text{LHS}})$. As a result, a set of decision and objective vectors pairs are generated using this technique as well.

4. Pareto-front selection. In this step, non-dominated solutions are selected among a set of all generated points using different methods in previous steps. The set is the union of three subsets including: 1. points on PF from the previous generation, (X, Y). 2. the estimated points by the trained model, (X_M, Y_M); as previously mentioned, after training the model, the approximate decision values of training data are evaluated by the fitness function to compute the true value of objectives. Therefore, in addition to PF from the previous iteration, the approximated points around the PF are appropriate set of candidates to cover the PF similar to a local search which covers the region around the current candidate solutions. 3. the resulted points from the generative techniques such as OBL and LHS as follows.

$$\begin{aligned}(X, Y) = \text{NDSorting}((X, Y) \cup (X_M, Y_M) \\ \cup (X_{\text{opI}}, Y_{\text{opI}}) \cup \cdots (X_{\text{opII}}, Y_{\text{opII}}) \cup (X_{\text{popII}}, Y_{\text{popII}}) \\ \cup (X_{\text{LHS}}, Y_{\text{LHS}})).\end{aligned} \tag{12}$$

The non-dominated solutions extracted from this step are considered as a new training set for the machine leaning algorithm to re-train it. Accordingly, the process of training the model and generating a well-distributed PF continues iteratively. At each iteration of the framework, by increasing the non-dominated points, more training data are provided to improve the quality of reverse mapping model. The overall structure of the MORM is illustrated in Fig. 3. As it is presented, the resulted candidate solution from an optimization process is fed into the ANN as training data. Simultaneously, other generative techniques including OBL and LHS are utilized to produce more objective and decision vectors. All generated points in objective space are mapped to decision space using the trained NN to get the approximate corresponding decision vectors. The predicted decision vectors are evaluated by the fitness function to compute the true objective values. Non-dominated solutions of a union set of new points are selected to re-train the NN to get a more accurate model. In fact, re-training ANN with new solutions is for improving the model but we can still stay with the same trained ANN if the training cost is not affordable. In other words, the proposed model follows the active learning however, it could be a passive one. Furthermore, most of the real-world optimization problems are many-objective with expensive objective functions which require a huge population size to generate as many as possible PF candidate solutions. However, the cost of fitness call of such problems is a serious barrier for evolving a large population. Hence, the cost of training ANN even with all non-dominated solutions is less than running a traditional many-objective optimization evolutionary algorithm with a huge population size. In fact, re-training of an ANN with a small set of samples is not time consuming.

By this iterative procedure, at the end of each iteration, a set of significant number of non-dominated solutions are added to PF. Finally, a well-distributed and dense PF can be resulted. The pseudocode of the algorithm is presented in Algorithm 1.

Experimental results and analysis

In this section, we explain the conducted experiments to assess the proposed method in terms of various performance metrics. In addition to comparison with the state-of-the-art many-objective optimization algorithms, the PF covering process of the proposed framework is investigated comprehensively.

Benchmark functions and implementation details

The practical application of the MORM is assessed in terms of various evaluation measures on a number of well-known many-objective optimization benchmarks [7] listed

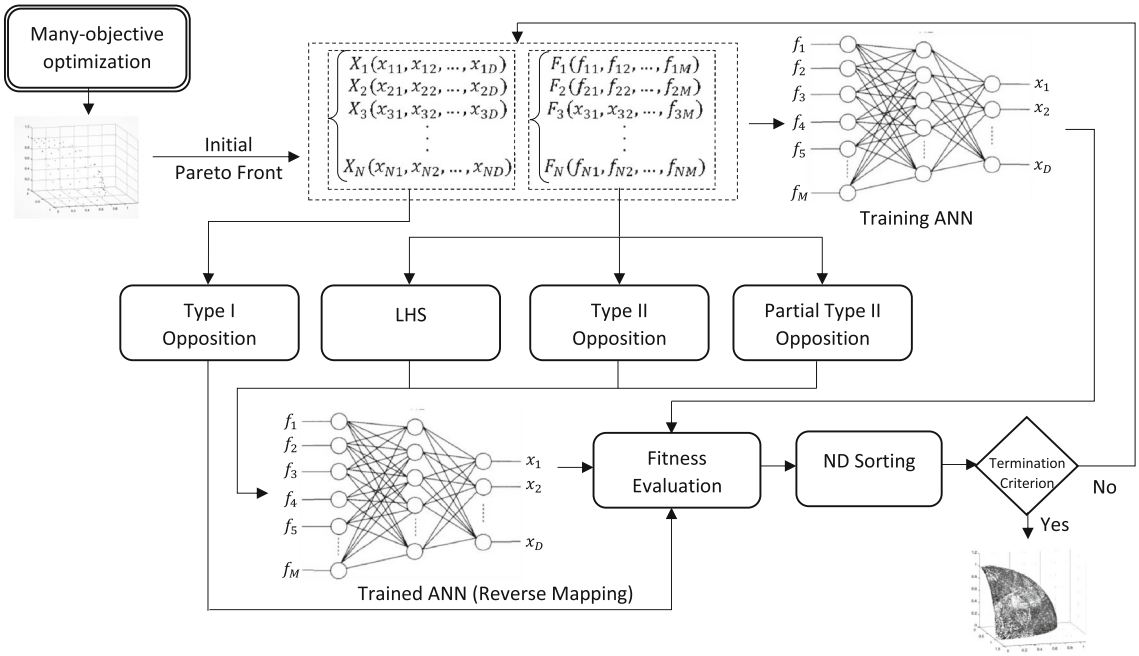


Fig. 3 The overall structure of the MORM. The sparse initial PF is the training data of the ANN. Several generative techniques including OBL and LHS are employed to add new candidate solutions to PF. The process is a an iterative procedure to cover the PF as much as required. ND sorting stands for Non-dominated sorting algorithm

in Table 1. The conducted experiments are for three and five-objectives optimization test sets. The number of decision variables is set according to the setting used in [7]. We excluded $MaF10$ because the optimization algorithm could not generate an initial PF close the true PF and consequently the reverse mapping procedure fails to construct a model on the resulted PF.

At the first step, NSGA-III [10] algorithm is applied to create the initial PF as the input of the proposed framework. The size of the population to run the NSGA-III is set to 100; consequently according to reference-line strategy in the NSGA-III, the number of solutions on the resulted PF is 91 for 3-objective problems, and 81 for 5-objective problems. An ANN model with the hidden layer size of 10 is trained using the fitness values of initial PF as the input and the decision variables values as the output of the network. The ratio of size of training, validation, and test sets for training of the ANN is set to 70%, 15%, and 15% of the entire data, respectively. The MSE is used as the performance function of the network. To handle the box-constraint for the generated candidate solutions, the infeasible values are replaced with average values of the corresponding dimension's boundaries.

Two categories of results are presented in this section. Firstly, the resulted solutions of MORM are represented by the visualization and numerical values. Moreover, the MORM is compared with three state-of-the-art many-objective optimization algorithms, namely NSGA-III, MOEA/D [48], and IM-MOEA [6] in terms of three well-known many-objective evaluation metrics including ratio of union PF, Hypervolume indicator (HV) [43], and inverted generational distance (IGD) [35]. Ratio of union PF computes the amount of contribution of each algorithm on resulted PF by conducting the NDS algorithm on them and then measuring the ratio of each. This measure compares the algorithms in terms of dominance. HV is a very popular indicator which evaluates many-objective algorithms in terms of the distribution of the PF and the closeness to true PF. HV indicator evaluates the diversity and convergence of a many-objective algorithm. It calculates the volume of M-dimensional space that is surrounded by a set of solution points (A) and a reference point $r = (r_1, r_2, \ldots, r_M)$ where M is the number of objectives of the problem. Therefore, the volume of the two-dimensional space which is surrounded by points on the obtained PF and r, is calculated as HV indicator. A reference point is a point with the worse values than nadir point. The measure is defined in Eq. 13.

$$\mathrm{HV}(A) = \mathrm{vol}\left(\bigcup_{a\in A}[f_1(a), r_1] \times [f_2(a), r_2] \times \cdots \times [f_M(a), r_M]\right), \tag{13}$$

where $a \in A$ is a point which all candidate solutions are weakly dominated by it. Larger values of HV indicates that

Table 1 Characteristics of benchmark problems

Test function	Properties	Dimension
MaF1	Linear	$M+9$
MaF2	Concave	$M+9$
MaF3	Convex, multimodal	$M+9$
MaF4	Concave, multimodal	$M+9$
MaF5	Convex, biased	$M+9$
MaF6	Concave, degenerate	$M+9$
MaF7	Mixed, disconnected, multimodal	$M+19$
MaF8	Linear, degenerated	2
MaF9	Linear, degenerated	2
MaF11	Convex, disconnected, nonseparable	$M+9$
MaF12	Concave, nonseperable, biased deceptive	$M+9$
MaF13	Concave, unimodal, nonseperable, degenerate	5

M is the number of objectives

input : $Prob$: Optimization problem, Max_It: Number of iterations
output: (X, Y): Non-dominated solutions

```
// Generating initial PF using an
   optimization algorithm
(X, Y) =
Many − objective_Optimization_Algorithm(Prob);
It = 0;
while It < Max_It do
    // Training a Model by the resulted PF as
       the inputs
    Training_Model(X, Y);
    X_M = Trained_Model(Y);
    Y_M = Y_Evaluation(X_M);
    // Generating points using Opposition
       type I
    X_opI = TypeI_Opposition(X);
    Y_opI = Y_Evaluation(X_opI);
    // Generating points using Opposition
       type II
    Y'_opII = TypeII_Opposition(Y);
    X_opI = Trained_ANN(Y'_opII);
    Y_opII = Y_Evaluation(X_opII);
    // Generating points using Partial
       Opposition type II
    Y'_popII = Partial_TypeII_Opposition(Y);
    X_popII = Trained_ANN(Y'_popII);
    Y_popII = Y_Evaluation(X_popII);
    // Generating points using LHS
    Y'_LHS = LHS(Y);
    X_LHS = Trained_ANN(Y'_LHS);
    Y_LHS = Y_Evaluation(X_LHS);
    // Selecting non-dominated solutions
    (X, Y) = NDSorting((X, Y) ∪ (X_M, Y_M) ∪ (X_opI, Y_opI) ∪
    ... (X_opII, Y_opII) ∪ (X_popII, Y_popII) ∪ (X_LHS, Y_LHS));
    It = It + 1
end
```

Algorithm 1: Pseudo-code of MORM. Other possible termination conditions such as HV value, number of PF solutions can be considered.

PF surrounds a wider space and it results in more diverse solutions and also more closer to optimal PF.

IGD measures the distance between the resulted PF and true PF^* which is calculated as follows:

$$\mathrm{IGD}(A) = \frac{\left(\sum_{i=1}^{PF^*} d_i^q\right)^{1/q}}{PF^*}, \tag{14}$$

where d_i^q is the Euclidean distance between a solution from PF* to its nearest individual in the resulted PF and $q = 2$. A smaller IGD value indicates a lower distance to true PF and consequently better performance. To calculate IGD, roughly 10,000 reference points on the PF of each benchmark function are sampled by Das and Dennis's approach [9]. Finally, because of the stochasticity of the algorithm, the experiments are conducted as 31 independent runs. The Wilcoxon statistical test [14] is applied to investigate the significance of the acquired results. By this way, the winner method is highlighted in each table of numerical results. All simulations are implemented using Matlab R2019a with Microsoft Windows 10 Enterprise 64-bit as the operating system on a PC with a AMD Ryzen Threadripper 1950X 16-Core Processor, 3750 Mhz.

Results and discussion

Figure 4 illustrates the resulted non-dominated solutions on PF for 3-objective $MaF2$ at each iteration of the process. The generated points using each component are presented using different colors. As it can be seen, the exploitation of each technique intensively increases the number of solutions on PF. Each technique discovers a portion of the PF and together they progressively cover the entire PF. Although, the number of solutions that each component can find is different, the effect of generated points in terms of filling the sparse region

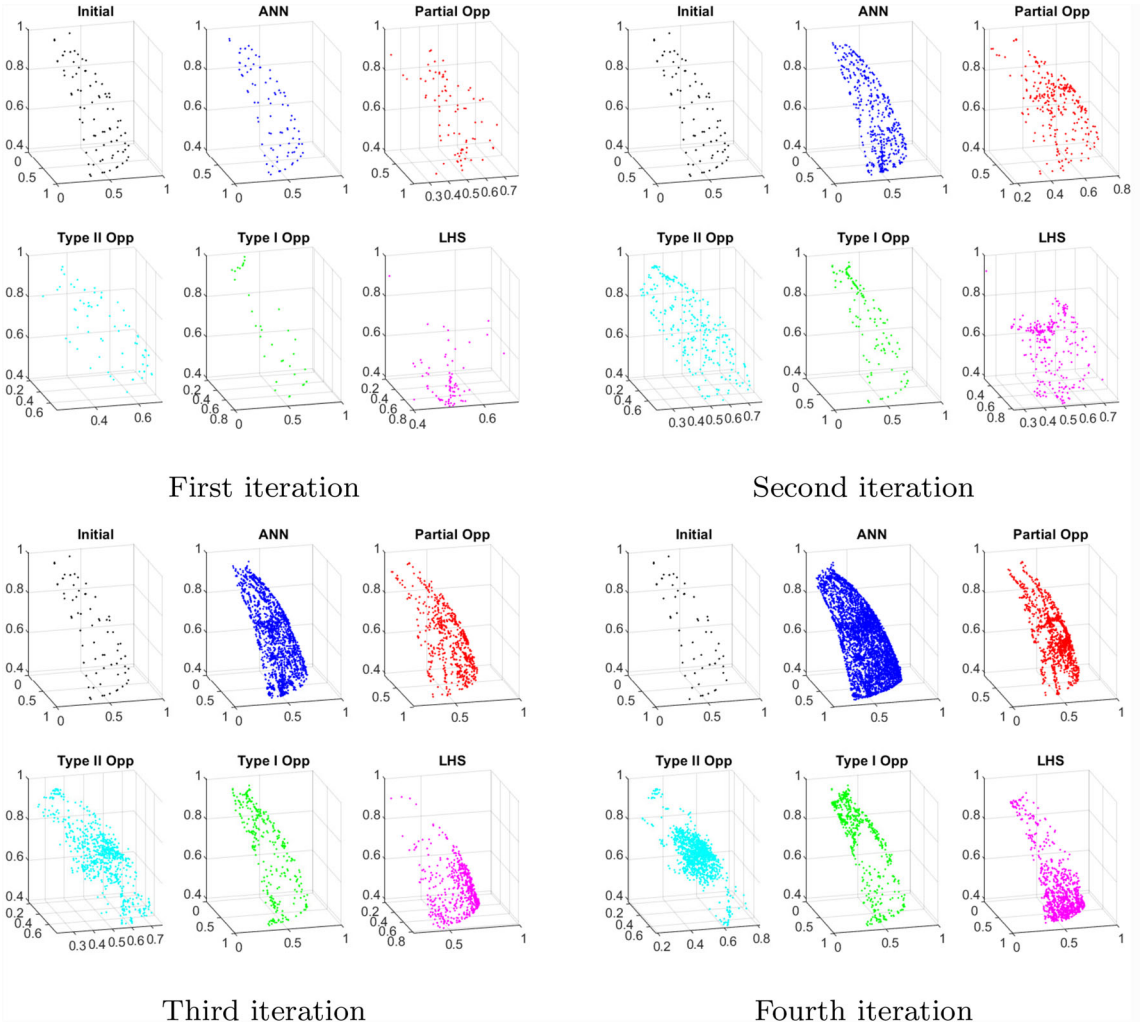


Fig. 4 Generated PF on MaF2 function for three objectives using each component during four consecutive iterations of the MORM. The generated points by each generative technique is presented in a different color. The PF is gradually covered during a number of iterations. According to results, the generated points by feeding the PF to ANN covers more region of PF rather than other techniques

and/or diversity improvement can be independent of their quantity. For MaF2, LHS and ANN can markedly generate more points to fill the surface of the PF. By contrast, opposition type I and type II produce a limited portion of the PF. Note that the presented points are non-dominated candidate solutions which are remained in each iteration. For instance, the initial points have been reduced from the first iteration to the second iteration because other techniques could generate solutions that dominate the initial points. However, the number of non-dominated solutions increases gradually. It is worth mentioning that each generative technique finds a specific portion of the PF according to the way it operates. For instance, ANN generates the points based on what is learned from the initial PF resulted using a state-of-the-art many-objective algorithm, thus the produced points are mainly around the initial PF, especially in the first iteration. However, generating more points in sparse regions using other techniques in the next iterations provides the ANN with more training data which leads to a well-trained network and well-distributed points.

Figure 5 represents the initial and final PF for some sample functions with three objectives resulted by the MORM. The initial PF is the non-dominated solutions obtained by the NSGA-III algorithm which is the input data for ANN to train the network for further process. After employing different generative techniques in four consecutive iterations, the final illustrated PF is achieved. As it is shown, the resulted non-dominated solutions could fill the sparse regions of the PF. However, the performance of the method is mainly affected by the initial PF because well-trained ANN yields more accurate candidate solutions for the optimization problem. Therefore, as it is presented, for some cases that the initial

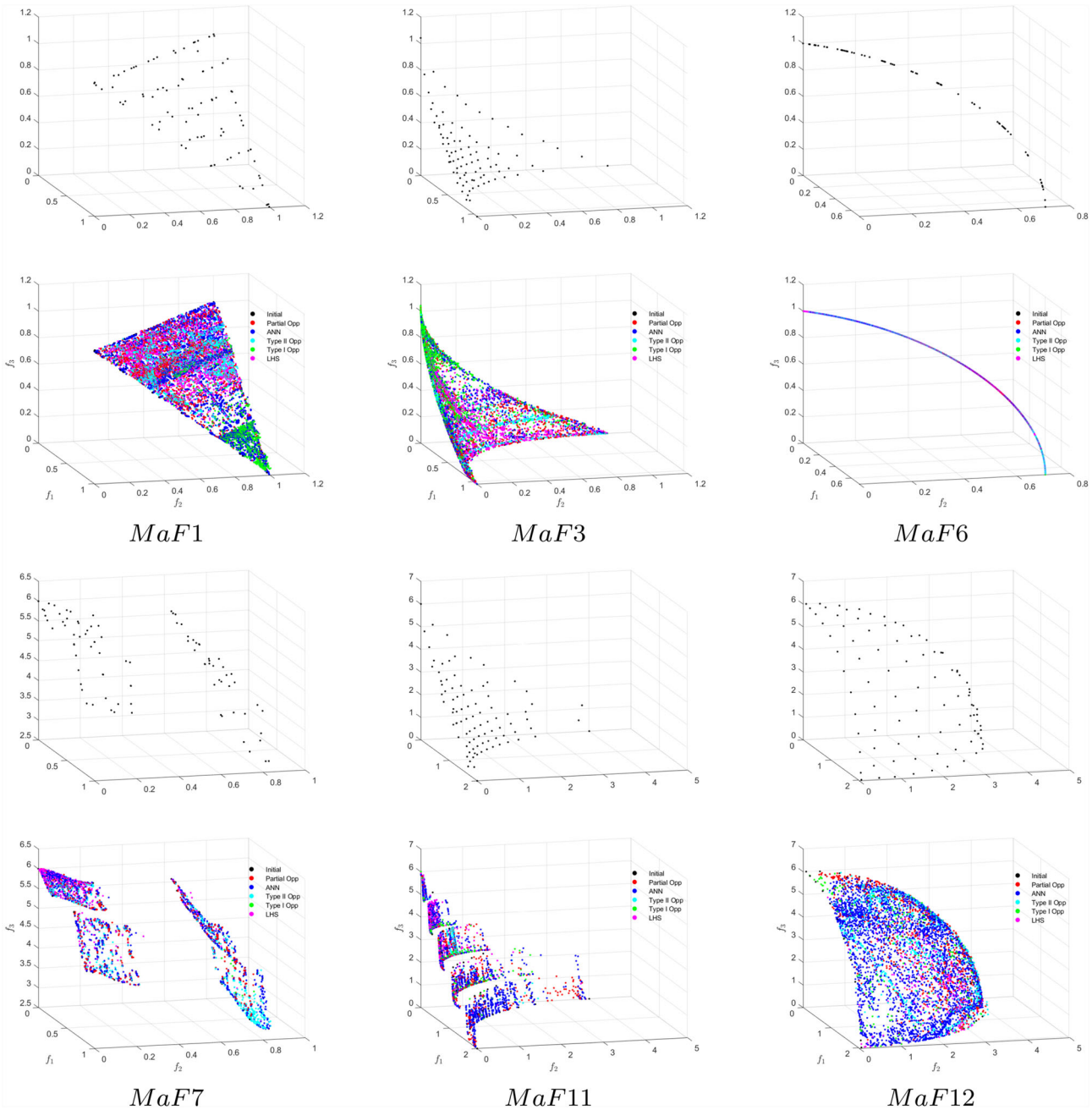


Fig. 5 Generated PF on samples of 3-objective benchmark functions. For each sample, the initial solutions generated by NSGA-III algorithm (top plots) and the final PF produced by the proposed method (bottom plots) are presented. The generated points by each generative technique is presented in a different color. The PF is accumulatively covered by different techniques during a number of iterations

optimization algorithm is not able to find solutions that cover the overall shape of the PF, the ANN cannot be trained efficiently and consequently, MORM may not fill the surface of the PF by the generated candidate solutions. Furthermore, Fig. 5 shows the distribution of non-dominated solutions resulted from different techniques. Each technique is able to find the solutions of a specific region of the PF. However, the ratio of the generated solutions is not identical for different techniques.

Figure 6 also illustrates the 3D-Radvis visualization [21] of the initial and final resulted PFs on some sample benchmark functions with five objectives. Similar to three objectives, the final PF is crucially dependent on the initial PF which provides the training data for ANN. During the process, all techniques performing in objective space require

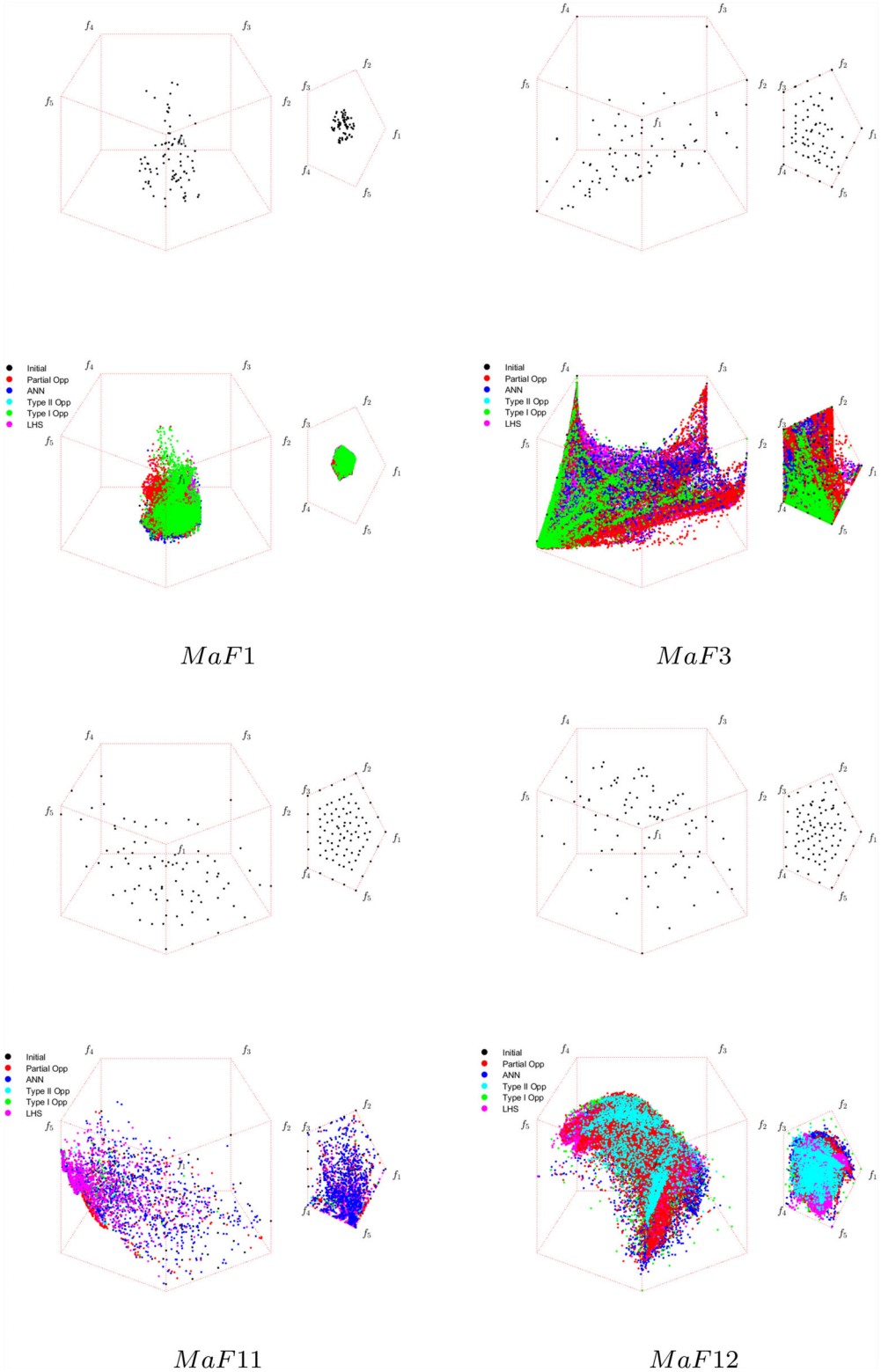


Fig. 6 3-D Radvis plots of Generated PF on samples of 5-objective benchmark functions. For each sample, the initial solutions generated by the NSGA-III algorithm (top plots) and the final PF produced by the MORM (bottom plots) are presented. The volume of generated points by each generative techniques is presented in a different color. The PF is accumulatively covered by different technique during a number of iterations

the ANN to estimate the value of variables in decision space. Consequently, the performance of ANN has a crucial role on the performance of the method. For instance, on MaF3, since the NSGA-III could not discover non-dominated points in a specific portion of the PF, i.e., the region between f_1 and f_2, the proposed method was not also able to search the corresponding region effectively and thus as it is shown, the number of discovered solutions is limited on the corresponding region.

In addition, Table 2 represents the numerical results of the proposed framework after four iterations for both 3- and 5-objective problems. The first column indicates the number of points on initial PF resulted from the NSGA-III, which the MORM starts the reverse mapping process from. Correspondingly, the number of final candidate solutions is given in the second column. Depending on the nature of the optimization problem and the shape of the PF, the number of generated candidate solutions is different. The maximum number of non-dominated solutions for 3-objective problems is 80,188 which is obtained for MaF8. This number for 5-objective benchmarks is 100,924 on MaF2. The MORM could increase the average number of candidate solutions on 3-objective PF from 91 to 19,773 whereas on 5-objective problems, the MORM reached 37,209 candidate solutions.

Each successive column shows the ratio of contribution of each component on the final PF. For instance, ANN has produced 39.48% and 31.69% of points on the resulted 3-objective and 5-objective PFs, respectively during the four iterations. As mentioned previously, ANN is trained at the beginning of each iteration using all non-dominated solutions which increase in every step; the ANN will be trained by more training data points and consequently it will be able to predict the decision variables with higher accuracy. On the other hand, the approximated decision values are able to reach new objective values on the PF close to previous points. This is the main reason why the ANN is in the first rank for all functions. In fact, the techniques such as LHS or OBL are able to explore the area around the PF while the ANN is a kind of local search to exploit the region close to the current PF. In other words, it is revealed that the ANN performs well on generating more points whereas other techniques cover more sparse regions. Results supports that OBL-based points lead to diversity and survive after non-dominated sorting to cover the sparse region of the PF. Along with the ratio of each components' contribution, the ratio of the remained points of the initial PF is also reported in the last column which constructs the lowest portion of the final PF. In overall, ANN has the higher contribution on finding the non-dominated solutions. Conversely, Type I Opposition could generate the lowest number of solutions among the employed techniques. The possible cause is that all other techniques explore the objective space to find more non-dominated solutions; therefore, they will achieve high probability of remarkable accomplishment. Conversely, type I opposition operates on decision space and correspondingly it may decrease the chance of selecting a non-dominated solution.

Well-distributed PF solutions using reference points

To ensure the diversity in resultant solutions, a predefined set of reference points can be used similar to that is proposed in NSGA-III. The reference points lead to widely distributed solutions because the reference points are widely distributed on the entire normalized hyperplane. NSGA-III uses Das and Dennis's [22] systematic approach to generate reference points on a normalized hyper-plane. The number of reference points (H) in an M-objective problem is defined by the following equation.

$$H = \binom{M + P - 1}{P}, \tag{15}$$

where P is the number of divisions along each objective. In our experiments, we define a predefined number of reference points to select a set of well-distributed candidate solutions among all generated non-dominated solutions. This technique produced a diverse set of non-dominated solution on the PF a diverse PF which makes the decision making more efficient. Figure 7 illustrates two samples of well-distributed PF with different number of reference points. Each reference point attracts a candidate solution having the minimum perpendicular distance with the corresponding reference line. As it is presented, increasing reference points cover more sparse region of the PF with adequate diversity. Moreover, among the selected points, there are generated points from different components of proposed methods indicating their contribution in producing diverse solutions. The number of total reference points are considered 500 and 1500 approximately; however, according to Eq. 15, the accurate values are the closest numbers which are 496 and 1485 for $M = 3$ and 495 and 1365 for $M = 5$, respectively.

Comparative results

To evaluate the MORM, we also compared it with the state-of-the-art algorithms such as NSGA-III, MOEA/D, and IM-MOEA. It is obvious that the maximum number of non-dominated solutions on a PF generated by a population-based many-objective algorithm is equal to the population size. Accordingly, to guarantee a fair comparison, for each benchmark function the competitors run with population size equal to the number of candidate solutions on the resulted PF from MORM. In addition, to run these algorithms, the total number of fitness calls set to that one required for four iterations of MORM plus the needed budget for running the optimization method to obtain the initial PF. Table 3 represents the

Table 2 Contribution of each generative component in final PF for 3-objective and 5-objective optimization problems

	Function	Initial	Final PF	ANN (%)	Type II opp (%)	Partial opp (%)	LHS (%)	Type I opp (%)	Remained initial (%)
$M = 3$	MaF1	91	15,313	36.27	19.5	16.51	16.63	10.6	0.5
	MaF2	91	9471	55.22	14.22	12.75	10.32	6.87	0.61
	MaF3	91	8650	20.39	14.1	13.36	19.17	32.89	0.08
	MaF4	91	65,354	21.22	21.73	22.51	23	11.52	0.02
	MaF5	91	3192	35.68	31.55	21.59	6.02	2.38	2.79
	MaF6	91	16,259	26.23	24.34	20.84	25.16	3	0.43
	MaF7	91	6437	46.39	25.32	17.71	9.1	0.17	1.3
	MaF8	91	80,188	20.39	20.31	20.24	20.47	18.51	0.08
	MaF9	91	21,934	27.41	22.23	22.53	16.53	10.96	0.34
	MaF11	91	3461	50.68	10.75	15.17	14.91	6.44	2.05
	MaF12	91	5301	63.2	14.41	12.19	4.66	4.4	1.15
	MaF13	91	1724	70.71	0	9.63	14.56	0	5.1
	Avg.	91	19,773	39.48	18.2	17.08	15.04	8.97	1.2
	Function	Initial	Final PF	ANN (%)	LHS (%)	Partial opp (%)	Type II opp (%)	Type I opp (%)	Remained initial (%)
$M = 5$	MaF1	85	43354	25.72	18.47	20.83	23.84	11.07	0.07
	MaF2	85	100,924	19.87	20.82	20.19	19.88	19.16	0.08
	MaF3	85	64,224	20.23	22.72	19.35	21.51	16.1	0.1
	MaF4	85	74,692	22.11	21.69	22.05	21.79	12.3	0.07
	MaF5	85	3013	39.1	10.39	26.12	13.14	8.63	2.62
	MaF6	85	5051	22.83	9.92	27.36	26.49	11.9	1.5
	MaF7	85	14,247	30.41	16.54	28.03	24.48	0.01	0.54
	MaF8	85	85,962	20.38	20.11	20.18	20.26	18.99	0.09
	MaF9	85	2940	60.75	35.48	1.16	0.03	0	2.59
	MaF11	85	2877	36.18	29.82	23.04	3.02	5.07	2.85
	MaF12	85	49,006	27.18	29.96	16.42	5.85	20.45	0.13
	MaF13	85	223	55.61	9.87	1.35	0	0	33.18
	Avg.	85	37,209	31.69	20.48	18.83	15.02	10.3	3.65

Two first columns represent the initial and final number of resulted non-dominated solutions. Successive columns indicate the portion of the final PF which is generated by the corresponding component. In addition, the portion of initial PF which is remained non-dominated after generating new points, shown in the last column

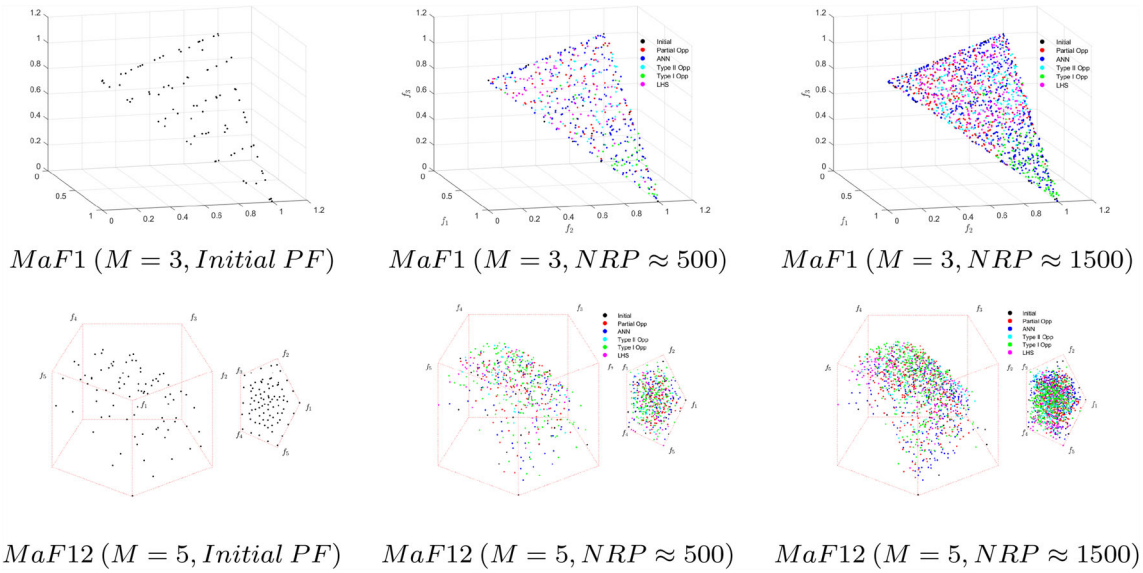


Fig. 7 Well-distributed PFs generated using reference points on two samples. For each sample, the initial solutions generated by the NSGA-III algorithm (left plots), the final PF produced by the proposed method and ≈500 reference points (middle plots), and ≈ 1500 reference points (right plots) are presented. NPR indicates the number of reference points

population size and NFC allocated for each benchmark function according to the number of points on final PF and the number of given NFC of MORM, respectively. As it is presented, the number of these parameters are different for 3- and 5-objective optimization problems. Considering these parameters for many-objective optimizations, the maximum number of iterations for the algorithm to evolve the population is four which is a very small value and consequently insufficient for the algorithm to reach a comparable PF. For instance, the population size for $MaF2$ is 9471 while the algorithm can call the fitness function 67,905 times. Therefore, at each iteration the competitor algorithms can iterate only seven times to evolve the population which is a very low value for the number of iterations for such a large population to get evolved sufficiently.

According to the aforementioned parameters, NSGA-III, MOEA/D, and IM-MOEA as competitors are applied on the optimization benchmark functions. For all algorithms, we use the PlatEMO framework [39]. The parameter setting in platform is based on the reference papers which are given in Table 4.

The results of all algorithms are presented in Tables 5 and 6. As it can be seen, the algorithms are compared with the MORM in terms of ratio of union PF, HV, and IGD. Union PF is a set of non-dominated solutions from the union of PFs resulted from three methods. The remaining non-dominated solutions from each method in the union PF is reported as a measure. From the Table 5, MORM has a higher contribution in the union PF in all optimization problems except 3-objective MaF1 in which MOEA/D performs better. Since

Table 3 Population size and allocated number of fitness call for state-of-the-art optimization algorithms (i.e., competitors)

	$M = 3$		$M = 5$	
	POP Size	NFC	POP Size	NFC
MaF1	15,313	69,655	43,354	103,310
MaF2	9471	67,905	100,924	148,040
MaF3	8650	707,40	64,224	115,580
MaF4	65,354	120,350	74,692	129,635
MaF5	3192	44,680	3013	53,870
MaF6	16,259	87,840	5051	68,725
MaF7	6437	87,105	14,247	102,245
MaF8	80,188	89,815	85,962	94,545
MaF9	21,934	37,540	2940	15,935
MaF11	3461	48,230	2877	53,120
MaF12	5301	58,390	49,006	119,840
MaF13	1724	22,355	223	17,425
Avg.	19,773	670,50	37,209	85,189

The population size is equal to the resultant number of non-dominated solutions by MORM. NFC represents the number of fitness calls

the solutions generated by the NSGA-III and IM-MOEA algorithm were dominated by the solutions form other algorithm for most of the test problems, its contribution is zero. Similarly, the difference between the ratio of contribution of MORM and MOEA/D is remarkably significant. From Table 6, HV and IGD also reveal the superiority of the MORM compared to other algorithms. On average, in terms of HV, the MORM surpasses the NSGA-III, MOEA/D, and on 10, 6,

Table 4 Parameter settings of the three algorithms in comparison

NSGA-III	
Population size	According to Table 3
Number of fitness calls	According to Table 3
Number of reference points	According to [10]
Generative operators	Genetic operators
MOEA/D	
Population size	According to Table 3
Number of fitness calls	According to Table 3
Decomposition method	Tchebychef
Number of neighbors	10
Generative operators	DE operators
Number of reference points	According to [48]
IM-MOEA	
Population size	According to Table 3
Number of fitness calls	According to Table 3
Number of reference vectors	10
Model group size	3

The population size and number of fitness calls are set based on what are obtained from proposed method according to Table 3

and 9 out of 12 of 3-objective functions, respectively. Similar HV results are acquired for 5 objectives on which the average value of HV achieved by MORM is 0.86 whereas NSGA-III, MOEA/D, and IM-MOEA could obtain a PF with HV value of 0.7, 0.65, and 0.8, respectively. Considering IGD values, MORM has obtained a PF with less distance to the true PF which yields lower values of IGD compared to competitors. Accordingly, except on 3-objective MaF9 and MaF13, and 5-objective $MaF6$, MORM outperforms other algorithms. For some functions, the large values of IGD indicates that resultant PF by competitors has a high distance of true PF and consequently they fail solve the problems. From our results, we can conclude that the MORM is able to generate many non-dominated solutions while state-of-the-art multi-objective algorithms fail to produce a well-distributed PF over a limited number of generations. In addition to the quantity, the generated solutions have the efficiency to increase the HV and to get closer to true PF according to IGD measure. As it is mentioned before, this is due to drawback of the multi-objective EAs, which require a large population size with numerous fitness calls to evolve such population. To fill the Pareto-front with a huge number of points, evolutionary algorithms require a large population size and correspondingly a huge number of fitness calls. The smaller number of fitness calls (in average) is needed compared to required budget for filling Pareto-front (i.e., generating non-dominated solutions with high accuracy) using an evolutionary algorithm. Therefore, all points are certainly required to be evaluated but as table shows this amount of fitness calls is not sufficient for a traditional multi-objective evolutionary algorithm lonely to fill the Pareto-front.

Table 5 Comparison on MORM, NSGA-III, MOEA/D, and IM-MOEA in terms of the ratio of union PF on 3-objective and 5-objective optimization problems

		MORM	NSGA-III	MOEA/D	IM-MOEA
$M = 3$	MaF1	35.71	0	**64.29**	0
	MaF2	**49.78**	0.01	38.20	12.02
	MaF3	**96.23**	0.10	3.67	0
	MaF4	**74.66**	0	25.33	0
	MaF5	**65.88**	0.06	31.62	2.44
	MaF6	**60.41**	0	39.59	0
	MaF7	**67.26**	0	32.59	0.15
	MaF8	**68.78**	0	31.22	0
	MaF9	**73.28**	0	26.67	0.05
	MaF11	**98.21**	0	1.79	0
	MaF12	**91.36**	0	6.03	2.61
	MaF13	**71.13**	0	13.17	15.70
	Avg.	**71.06**	0.01	26.18	2.75
	$w/t/l$		12/0/0	11/0/1	12/0/0
$M = 5$	MaF1	**53**	0.01	46.99	0
	MaF2	**74.73**	8.76	8.66	7.85
	MaF3	**96.23**	0.10	3.67	0
	MaF4	**99.99**	0	0.01	0
	MaF5	**48.82**	6.42	24.69	20.07
	MaF6	**57.13**	0	42.87	0
	MaF7	**54.75**	0	45.25	0
	MaF8	**59.50**	0.30	39.61	0.59
	MaF9	**59.30**	0	40.68	0.02
	MaF11	**55.87**	0	43.29	0.84
	MaF12	**59.50**	0.30	39.61	0.59
	MaF13	**48.37**	0	20.30	31.33
	Avg.	**63.93**	1.32	29.64	5.11
	$w/t/l$		12/0/0	12/0/0	12/0/0

The two last rows indicate the average values and the number of wins/ties/loses of MORM compared to competitors in terms of each evaluation measure. Bold values show the winner algorithm

Effectiveness on real-world problems

In this section, MORM has been employed on two cases of many-objective real-world problems to investigate its effectiveness.

Vehicle crashworthiness design problem

In automotive industry, there are a variety of important requirement of crashworthiness design to develop high-quality and low-cost products. To address these criteria, a

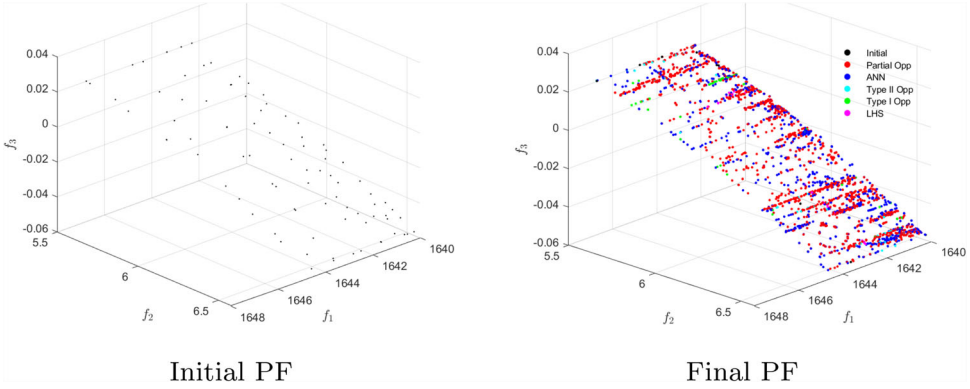
Table 6 Comparison on MORM, NSGA-III, MOEA/D, and IM-MOEA in terms of HV and IGD on 3-objective and 5-objective optimization problems

		HV				IGD			
		MORM	NSGA-III	MOEA/D	IM-MOEA	MORM	NSGA-III	MOEA/D	IM-MOEA
$M = 3$	MaF1	**0.65**	0.49	0.65	0.68	**0.01**	0.21	**0.01**	0.11
	MaF2	**0.36**	0.35	**0.36**	**0.36**	**0.01**	0.02	**0.01**	**0.01**
	MaF3	**1**	**1**	**1**	1	**0.01**	2.55E+04	0.08	3.60E+04
	MaF4	**1**	0.95	**1**	0.95	**0.05**	621.73	0.65	711.63
	MaF5	**0.93**	0.77	**0.93**	**0.93**	**0**	1.91	0.23	0.53
	MaF6	**1**	0.79	0.99	1	**0**	4.99	0	3.52
	MaF7	**0.78**	0.52	0.75	0.77	**0.02**	3.47	0.03	0.06
	MaF8	**0.99**	0	**0.99**	0.01	**0.01**	76.13	0.02	63.42
	MaF9	**1**	**1**	**1**	**1**	0.96	31.98	1.03	**0.89**
	MaF11	**0.96**	0.75	0.84	0.88	**0.14**	0.45	1.70	0.32
	MaF12	**0.65**	0.49	0.61	0.61	**0.07**	0.41	0.13	0.15
	MaF13	**0.99**	0.92	0.94	1	0.49	0.82	0.31	**0.10**
	Avg.	**0.86**	0.67	0.84	0.77	**0.15**	2188.34	0.35	3061.42
	w/t/l		10/2/0	6/6/0	9/3/0		12/0/0	10/2/0	9/2/1
$M = 5$	MaF1	0.29	0.17	**0.34**	0.26	**0.06**	0.29	0.03	0.32
	MaF2	**0.36**	**0.36**	**0.36**	0.38	**0.01**	0.03	0.02	0.02
	MaF3	**1**	**1**	**1**	**1**	**1.41**	6.69E+04	4.31E+04	3.60E+04
	MaF4	**1**	0.85	0.84	0.86	**1.48**	2318.52	2296.92	2253.84
	MaF5	**1**	0.98	0.58	0.99	**0.06**	4.23	11.26	2.77
	MaF6	0.96	0.71	0.97	**1**	0.10	3.65	**0.01**	1.50
	MaF7	**0.86**	0.63	0.11	0.79	**0.09**	3.79	0.66	1.16
	MaF8	**0.96**	0.27	0	0.71	**0.20**	26.63	111.91	0.98
	MaF9	**1**	0.99	**1**	**1**	**0.68**	81.78	0.93	1.19
	MaF11	**0.99**	0.79	0.91	0.89	**0.16**	0.65	4.62	0.44
	MaF12	**0.87**	0.69	0.84	0.71	**0.33**	1.10	0.68	0.98
	MaF13	**1**	0.95	0.80	**1**	**0.24**	1.25	1.06	1.06
	Avg.	**0.86**	0.70	0.65	0.80	**0.40**	5775.05	3794.34	3185.05
	$w/t/l$		10/2/0	7/4/1	8/4/0		12/0/0	11/0/1	12/0/0

The two last rows indicate the average values and the number of wins/ties/loses of MORM compared to competitors in terms of each evaluation measure. Higher values of the HV are desirable whereas for IGD, the lower value represents the outperformance. Bold values show the winner algorithm

multi-objective optimization problem can be defined. In [24], three objectives including the mass of the vehicle, an integration of collision acceleration, and the toe board intrusion are considered as design objectives. These objectives are formulated using the thickness of five reinforced members around the frontal structure as design variables (Eq. 16). The mass of the vehicle (f_1) is minimized for the consideration of lightweight. In addition, the minimum integration of collision acceleration (f_2) reflects the worst scenario of acceleration-induced biomechanical damage of occupants. Finally, to minimize the mechanical injury, the toe board intrusion in the "offset-frontal crash" (f_3) should be minimized:

$$\begin{aligned}
\min\ f_1(\boldsymbol{x}) &= 1640.2823 + 2.3573285x_1 + 2.3220035x_2 \\
&\quad +4.5688768x_3 + 7.7213633x_4 \\
&\quad +4.4559504x5 \\
\min\ f_2(\boldsymbol{x}) &= 6.5856 + 1.15x_1 - 1.0427x_2 + 0.9738x_3 \\
&\quad +0.8364x_4 - 0.3695x_1x_4 \\
&\quad +0.0861x_1x_5 + 0.3628x_2x_4 - 0.1106x_1^2 \\
&\quad -0.3437x_3^2 + 0.1764x_4^2 \\
\min\ f_3(\boldsymbol{x}) &= -0.0551 + 0.0181x_1 + 0.1024x_2 \\
&\quad +0.0421x_3 - 073x_1x_2 \\
&\quad +0.024x_2x_3 - 0.0118x_2x_4 - 0.0204x_3x_4 \\
&\quad -08x_3x_5 - 0.0241x_2^2 \\
&\quad +0.0109x_4^2, \qquad (16)
\end{aligned}$$



Initial PF Final PF

Fig. 8 Initial PF with 91 points and final PF with 6,325 points resulted from applying MORM for vehicle crashworthiness design problem

where $x_i \in [1, 3]$ for $i \in \{1, 2, 3, 4, 5\}$ are the thickness of five reinforced members around the frontal structure which could significantly affect the crash safety. More details can be find in [24]. An initial PF is obtained using NSGA-III algorithm to apply the MORM. After four iterations, the initial PF with 91 points is filled with 6,325 non-dominated solutions. Figure 8 represents the initial and final PFs resulted from applying MORM. As it can be seen, the generative techniques collaborated to result a well-distributed PF.

Rocket injector design problem

Design of injectors is an important element to develop technologies to make the next generation launch systems safer, more affordable, and more reliable. In [42], the design considerations are guided by three design objectives, namely, the minimum temperature on the injector face (f_1), the length of the combustion zone (f_2), and the temperature on the oxidizer post tip (f_3). Shorter combustion lengths account for better performing designs while lower temperatures would indicate a design that had longer life due to decreased thermal strain. To satisfy theses goals, three design variables are selected, namely the angle at which the hydrogen is directed toward the oxidizer (x_1), the change in hydrogen flow area from the baseline (x_2), the change in oxygen flow area from the baseline (x_3), and the oxidizer post tip thickness (x_4). Eq. 17 indicates the formulation of three objectives based on the four variables.

$$
\begin{aligned}
\min\ f_1(\boldsymbol{x}) &= 0.692 + 0.477x1 - 0.687x_2 - 0.080x_3 \\
&\quad -0.0650x_4 - 0.167x1x1 \\
&\quad -0.0129x_2x_1 + 0.0796x_2x_2 - 0.0634x_3x_1 \\
&\quad -0.0257x_3x_2 + 0.0877x_3x_3 \\
&\quad -0.0521x_4x_1 + 0156x_4x_2 \\
&\quad +0198x_4x_3 + 0.0184x_4x_4 \\
\min\ f_2(\boldsymbol{x}) &= 0.153 - 0.322x_1 + 0.396x_2 + 0.424x_3 \\
&\quad +0.0226x_4 + 0.175x_1x_1 \\
&\quad +0.0185x_2x_1 - 0.0701x_2x_2 - 0.251x_3x_1 \\
&\quad +0.179x_3x_2 + 0.0150x_3x_3 \\
&\quad +0.0134x_4x_1 + 0.0296x_4x_2 \\
&\quad +0.0752x_4x_3 + 0.0192x_4x_4 \\
\min\ f_3(\boldsymbol{x}) &= 0.370 - 0.205x_1 + 0.0307x_2 \\
&\quad +0.108x_3 + 1.019x_4 - 0.135x_1x_1 \\
&\quad +0.0141x_2x_1 + 0.998x_2x_2 + 0.208x_3x_1 \\
&\quad -0.0301x_3x_2 - 0.226x_3x_3 \\
&\quad +0.353x_4x_1 - 0.0497x_4x_3 - 0.423x_4x_4 \\
&\quad +0.202x_2x_1x_1 - 0.281x_3x_1x_1 \\
&\quad -0.342x_2x_2x_1 - 0.245x_2x_2x_3 \\
&\quad +0.281x_3x_3x_2 - 0.184x_4x_4x_1 \\
&\quad -0.281x_2x_1x_3, \qquad (17)
\end{aligned}
$$

where $x_i \in [0, 1]$ for each $i \in \{1, 2, 3, 4\}$.

An initial PF is obtained using NSGA-III algorithm to apply the MORM. After four iterations, the initial PF with 91 points is filled with 2664 non-dominated solutions. Figure 9 represents the initial and final PFs resulted from applying MORM. As it can be seen, the generative techniques collaborated to result a well-distributed PF.

Furthermore, the comparison between the MORM and other algorithms for both problems is given in Table 7. As it is showed, the HV value of resultant PF using MORM is 0.85 and 0.73 for Vehicle Crashworthiness and Rocket Injector, respectively, while other algorithms could reach to a lower HV value. Thus, the effectiveness of proposed method on real-world problems is significant.

Concluding remarks

Every many-objective optimization algorithm delivers a restricted amount of solutions as the output of the optimization process. However, the well-covered PF is generally desired for a variety of applications. In this paper, we have introduced a kind of collaboration between machine learning and many-objective optimization algorithms to increase

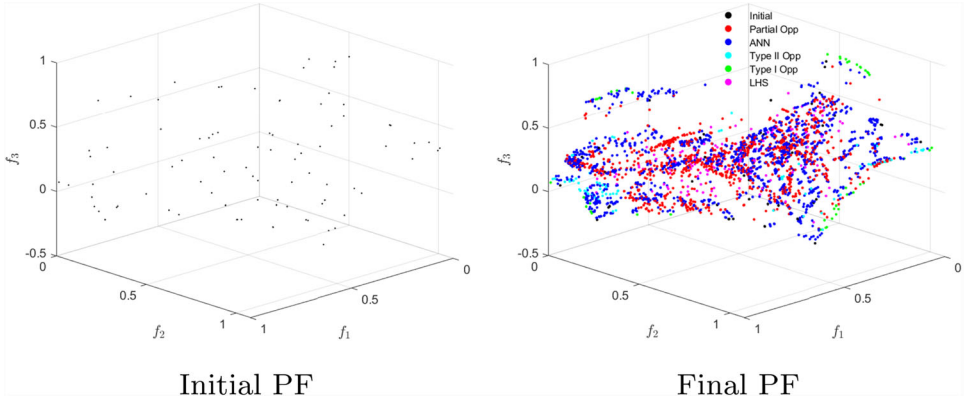


Fig. 9 Initial PF with 91 points and final PF with 2664 resulted from applying MORM for rocket injector design problem

Table 7 HV values of MORM and competitor algorithms for real-world problems

	MORM	NSGA-III	MOEA/D	IM-MOEA
Vehicle crashworthiness	**0.86**	0.74	0.82	0.78
Rocket injector	**0.73**	0.63	0.66	0.68

Bold values show the winner algorithm

the non-dominated solutions by an inexpensive approach. A many-objective optimization algorithm generates an initial PF which is the input of a learning model. The trained model maps the objective space points to decision space. This framework allows us to generate the many solutions by generative techniques in objective space and then find their decision variables using the trained model. Each component covers some regions of the PF and it consequently leads to a well covered PF. To ensure the diversity of the generated solutions, we used a set of reference points to select a predefined number of candidate solutions among all generated points. Compared to the initial PF solutions obtained by an optimization algorithm, the proposed method, MORM, can produce the resultant covered PF by only a few iterations. The conducted experiments on well-known many-objective benchmarks have demonstrated the effectiveness of the MORM in terms of several many-objective optimization assessment measures. The comparative results between the proposed scheme and the state-of-the-art many-objective optimization algorithms including NSGA-III, MOEA/D, and IM-MOEA have revealed that the surface of the generated PF by a collaboration of machine learning is covered by a cloud of non-dominated solutions. It has been presented that each generative technique covers a portion of the PF resulting in accumulatively a well-covered PF. However, generating a similar PF using many-objective optimization algorithms require a significantly high budget which is not applicable especially for expensive optimization problems. Generating a low cost PF solutions without overloading of the optimizer seems a very promising direction. This novelty is achieved as a fruitful result of collaboration between machine learning and optimization algorithms, especially when the problem is many-objective and the proposed scheme is fully independent from the family of optimizer, classical or meta-heuristic. However, since the population-based optimizers obtain a PF using the individuals' collaboration, they probably are to offer a much better input set (i.e., initial PF) to our proposed method whereas this property is lacking in single-solution based methods, where they build PF by hitting it point-by-point. In addition to benchmarks, MORM is evaluated on two well-known real-world problems . As future works, using the proposed approach in an interactive the optimization framework can improve the efficiency of the method. In addition, using the reference points can be embedded in optimization process to generate non-dominated solutions on desired regions of the PF.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Adra SF, Hamody AI, Griffin I, Fleming PJ (2005) A hybrid multi-objective evolutionary algorithm using an inverse neural network for aircraft control system design. In: 2005 IEEE congress on evolutionary computation, IEEE, vol 1, pp 1–8
2. Bechikh S, Elarbi M, Said LB (2017) Many-objective optimization using evolutionary algorithms: a survey. In: Recent advances in evolutionary multi-objective optimization, Springer, pp 105–137
3. Bidgoli AA, Rahnamayan S (2019) An innovization-based model to approximate geometric parameters of solar chimney power plant for desired efficiency and output power. In: 2019 International conference on power generation systems and renewable energy technologies (PGSRET), IEEE, pp 1–8
4. Bisbo MK, Hammer B (2020) Efficient global structure optimization with a machine-learned surrogate model. Phys Rev Lett 124(8):086102
5. Chen T, Tang K, Chen G, Yao X (2012) A large population size can be unhelpful in evolutionary algorithms. Theoret Comput Sci 436:54–70
6. Cheng R, Jin Y, Narukawa K, Sendhoff B (2015) A multiobjective evolutionary algorithm using gaussian process-based inverse modeling. IEEE Trans Evol Comput 19(6):838–856
7. Cheng R, Li M, Tian Y, Zhang X, Yang S, Jin Y, Yao X (2017) A benchmark test suite for evolutionary many-objective optimization. Complex Intell Syst 3(1):67–81
8. Cho S, Kim M, Lyu B, Moon I (2021) Optimization of an explosive waste incinerator via an artificial neural network surrogate model. Chem Eng J 407:126659
9. Das I, Dennis JE (1998) Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. SIAM J Optim 8(3):631–657
10. Deb K, Jain H (2013) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. IEEE Trans Evol Comput 18(4):577–601
11. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
12. Deb K, Bandaru S, Greiner D, Gaspar-Cunha A, Tutum CC (2014) An integrated approach to automated innovization for discovering useful design principles: case studies from engineering. Appl Soft Comput 15:42–56
13. Deb K, Srinivasan A (2006) Innovization: innovating design principles through optimization. In: Proceedings of the 8th annual conference on genetic and evolutionary computation, pp 1629–1636
14. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
15. Farias LR, Arajo AF (2021) Im-moead: an inverse modeling multi-objective evolutionary algorithm based on decomposition. In: 2021 IEEE international conference on systems, man, and cybernetics (SMC), IEEE, pp 462–467
16. Gaspar-Cunha A, Vieira A (2005) A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. Int J Comput Syst Signals 6(1):18–36
17. Gaspar-Cunha A, Vieira A et al (2004) A hybrid multi-objective evolutionary algorithm using an inverse neural network. In: Hybrid metaheuristics, Citeseer, pp 25–30
18. Gutiérrez-Antonio C, Briones-Ramírez A (2010) Speeding up a multiobjective genetic algorithm with constraints through artificial neuronal networks. In: Computer aided chemical engineering, vol 28. Elsevier, pp 391–396
19. He Z, Yen GG (2015) Many-objective evolutionary algorithm: objective space reduction and diversity improvement. IEEE Trans Evol Comput 20(1):145–160
20. He C, Cheng R, Yazdani D (2020) Adaptive offspring generation for evolutionary large-scale multiobjective optimization. IEEE Trans Syst Man Cybernet Syst
21. Ibrahim A, Rahnamayan S, Martin MV, Deb K (2016) 3d-radvis: Visualization of pareto front in many-objective optimization. In: 2016 IEEE congress on evolutionary computation (CEC), IEEE, pp 736–745
22. Indraneel D (1998) Normal-boundary intersection: a new method for generating the pareto surface in nonlinear multicriteria optimization problems. Soc Ind Appl Math 8(3):631–657
23. Kobayashi K, Hiroyasu T, Miki M (2007) Mechanism of multi-objective genetic algorithm for maintaining the solution diversity using neural network. In: International conference on evolutionary multi-criterion optimization. Springer, pp 216–226
24. Liao X, Li Q, Yang X, Zhang W, Li W (2008) Multiobjective optimization for crash safety design of vehicles using stepwise regression model. Struct Multidiscip Optim 35(6):561–569
25. Liu T, Li X, Tan L, Song S (2021) An incremental-learning model-based multiobjective estimation of distribution algorithm. Inf Sci 569:430–449
26. Lu Z, Whalen I, Dhebar Y, Deb K, Goodman ED, Banzhaf W, Boddeti VN (2020) Multiobjective evolutionary design of deep convolutional neural networks for image classification. IEEE Trans Evol Comput 25(2):277–291
27. Lye KO, Mishra S, Ray D, Chandrashekar P (2021) Iterative surrogate model optimization (ismo): an active learning algorithm for pde constrained optimization with deep neural networks. Comput Methods Appl Mech Eng 374:113575
28. Mahdavi S, Rahnamayan S, Deb K (2018) Opposition based learning: a literature review. Swarm Evol Comput 39:1–23
29. Mahdavi S, Rahnamayan S, Deb K (2016) Partial opposition-based learning using current best candidate solution. In: 2016 IEEE symposium series on computational intelligence (SSCI), IEEE, pp 1–7
30. Mittal S, Saxena DK, Deb K (2020) Learning-based multi-objective optimization through ann-assisted online innovization. In: Proceedings of the 2020 genetic and evolutionary computation conference companion, pp 171–172
31. Nguyen AQ, Sutton AM, Neumann F (2015) Population size matters: rigorous runtime results for maximizing the hypervolume indicator. Theoret Comput Sci 561:24–36
32. Park WJ, Park JB (2018) History and application of artificial neural networks in dentistry. Eur J Dent 12(04):594–601
33. Qi C, Chen Q, Kim SS (2020) Integrated and intelligent design framework for cemented paste backfill: a combination of robust machine learning modelling and multi-objective optimization. Miner Eng 155:106422
34. Rahnamayan S, Tizhoosh HR, Salama MM (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79
35. Rodríguez Villalobos CA, Coello Coello CA (2012) A new multi-objective evolutionary algorithm based on a performance assessment indicator. In: Proceedings of the 14th annual conference on Genetic and evolutionary computation, pp 505–512
36. Salehinejad H, Rahnamayan S, Tizhoosh HR (2014) Type-II opposition-based differential evolution. In: 2014 IEEE congress on evolutionary computation (CEC), IEEE, pp 1768–1775
37. Schweidtmann AM, Mitsos A (2019) Deterministic global optimization with artificial neural networks embedded. J Optim Theory Appl 180(3):925–948
38. Steponavice I, Shirazi-Manesh M, Hyndman RJ, Smith-Miles K, Villanova L (2016) On sampling methods for costly multi-objective black-box optimization. In: Advances in stochastic and deterministic global optimization. Springer, pp 273–296

39. Tian Y, Cheng R, Zhang X, Jin Y (2017) Platemo: a matlab platform for evolutionary multi-objective optimization. IEEE Comput Intell Mag 12(4):73–87
40. Tian Y, Zheng X, Zhang X, Jin Y (2019) Efficient large-scale multiobjective optimization based on a competitive swarm optimizer. IEEE Trans Cybern 50(8):3696–3708
41. Trivedi A, Srinivasan D, Sanyal K, Ghosh A (2017) A survey of multiobjective evolutionary algorithms based on decomposition. IEEE Trans Evol Comput 21(3):440–462
42. Vaidyanathan R, Tucker K, Papila N, Shyy W (2003) Cfd-based design optimization for single element rocket injector. In: 41st Aerospace sciences meeting and exhibit, p 296
43. While L, Hingston P, Barone L, Huband S (2006) A faster algorithm for calculating hypervolume. IEEE Trans Evol Comput 10(1):29–38
44. Xue X, Jiang C, Wang H, Tsai PW, Mao G, Zhu H (2021) An improved multi-objective evolutionary optimization algorithm with inverse model for matching sensor ontologies. Soft Comput 25(18):12227–12240
45. Yegnanarayana B (2009) Artificial neural networks. PHI Learning Pvt. Ltd., New Delhi
46. Yi JH, Xing LN, Wang GG, Dong J, Vasilakos AV, Alavi AH, Wang L (2020) Behavior of crossover operators in nsga-iii for large-scale optimization problems. Inf Sci 509:470–487
47. You J, Ampomah W, Sun Q (2020) Development and application of a machine learning based multi-objective optimization workflow for co2-eor projects. Fuel 264:116758
48. Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
49. Zhang H, Ding J, Jiang M, Tan KC, Chai T (2021) Inverse gaussian process modeling for evolutionary dynamic multiobjective optimization. IEEE Trans Cybern
50. Zhao B, Lu H, Chen S, Liu J, Wu D (2017) Convolutional neural networks for time series classification. J Syst Eng Electron 28(1):162–169