

Modified GUESS Approach: An Efficient Interactive Multi-Criterion Decision-Making Procedure Following Machine Decision-Makers

COIN Report 2026004

Deepanshu Yadav
Michigan State University
East Lansing, MI, USA
deepansh@msu.edu

Kalyanmoy Deb
Michigan State University
East Lansing, MI, USA
kdeb@msu.edu

Abstract—Interactive multi-criterion decision-making (iMCDM) procedures generally incorporate the human decision maker’s (DM’s) preferences to iteratively compute preferred Pareto-optimal (PO) solutions. DM preferences often involve objective classification indicating their improvement, relaxation, satisfaction, and indifference at the current PO solution. In addition, the DM provides one or more preference information in the form of a reference point, objective weights, and reference direction. These preferences are used to reformulate the original multi-objective optimization (MOO) problem into a single-objective optimization problem using the achievement scalarization function (ASF). The iterative preferences are then used to solve the ASF problem to compute a preferred PO solution. In this paper, we propose a modified version of a well-known iMCDM procedure: GUESS. In order to compare the modified GUESS with the original GUESS, we first propose a Machine Learning (ML)–based decision-maker (Machine-DM), which uses two Artificial Neural Networks (ANNs) to replace human DM steps in GUESS. The two trained ANNs predict the objective class and bounding parameters of objectives from the decision variable vector, without explicit information on the location and value of the preferred point, thereby enabling a fair performance evaluation. Following the Machine-DM approach, three variants of GUESS: MachDM-GUESS (original), MachDM-GUESS-I and MachDM-GUESS-II (modified), are developed and compared using newly proposed performance metrics. The implementation of Machine-DM with the original and modified GUESS procedures reveals superiority of one of the modified GUESS procedures. Moreover, the study conducted in this paper opens up new avenues for benchmarking existing iMCDM procedures and proposing new and competitive ones, paving the way and encouraging EMO and MCDM researchers to conduct more such studies.

Index Terms—Interactive Multi-criteria Decision-making, Machine Learning, Evolutionary Multi-objective optimization, Preference Learning, Benchmarking.

I. INTRODUCTION

REAL-world applications often consist of multiple and conflicting objectives to be simultaneously optimized, subject to a number of (in)equality constraints. Such applications are formulated as multi(many)-objective optimization (M(a)OO) problems. Evolutionary multi-objective optimization (EMO) algorithms have been primarily focused to solve

M(a)OO to obtain a well-distributed and diverse set of Pareto-optimal (PO) solutions [1]–[3]. NSGA-II [4], NSGA-III [5], MOEA/D [6], and C-TAEA [7] among others are well-known EMO algorithms to compute near Pareto Front (PF) solutions.

However, from implementation perspective, a decision-maker (DM) has to pick a single solution from the non-dominated (ND) solutions. This decision-making task, known as multi-criteria decision-making (MCDM) procedure, requires DM to incorporate preference information, which is then used to convert a MOO problem into a scalarized single-objective optimization problem [8], [9]. Based on the order of DM’s preference incorporation and performing optimization, MCDM methods are classified into three broad categories: (i) *a priori* methods (ii) *a posteriori* methods, and (iii) *interactive* methods. In *a priori* methods, DM preference is first obtained to formulate a scalarization problem, which is then optimized. In *a posteriori* methods, first a set of ND solutions are found using MOO and then DM’s preference information is used to pick the most preferred one. In *interactive* methods, DM’s preference incorporation and optimization are conducted iteratively, in which DM can modify their preferences in each iteration to formulate and solve a new scalarized function, till the most preferred solution is obtained [10].

The interactive methods are systemic procedures which allow DMs to elicit their preferences in various ways including objective weights, reference point, reference direction, aspiration level, objective classification, among others [11]. These flexibility allow DMs to choose an ASF or their augmented version (AASF), and then formulate and solve a scalarized problem [8] to obtain preferred solutions, iteratively. Using different choices of DM preference and (A)ASF, a number of interactive MCDM (iMCDM) procedures have been developed in the MCDM literature. The existing and well-known iMCDM procedures includes the Step Method (STEM) [12], Satisficing Trade-off Method (STOM) [13], Pareto Race [14], NIMBUS [15], and GUESS [16], among others [17].

In this paper, we propose two modified and efficient versions of the GUESS procedure. For a fair performance evaluation

of existing GUESS procedure with the modified ones, we first introduce the concept of Machine Learning-based DM (Machine-DM). In principle, Machine-DM replaces the human DM steps in the existing GUESS procedure with two Artificial Neural Networks (ANNs) and a preference evaluator. The proposed Machine-DM approach is generalizable to a broad class of iMCDM procedures including NIMBUS, STEM, and STOM, among others. The novelty and key contributions of the paper is outlined as follows:

- (1) Introduction to Machine Learning-based Decision-Maker,
- (2) Modified GUESS approach by systematic computation of reference points through updated objective bounds,
- (3) Performance evaluation metric for iMCDM procedures,
- (4) Comparison of GUESS approach with proposed two variants of modified GUESS method using Machine-DM.

The rest of the paper is organized as follows. Section II provides background on interactive multi-criterion decision-making procedures. Section III and IV discuss MachineDM approach and MachDM-GUESS procedure, respectively. Section V discusses the proposed approach for computing objective bounds and reference points. Simulation results are detailed in Section VI followed by conclusion in Section VII. The data and source codes for executing pre-trained ANNs used in this study are available on [GitHub](#).

II. BACKGROUND ON iMCDM PROCEDURES

Consider the following constrained MOO problem:

$$\begin{aligned}
& \text{Minimize} && (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_M(\mathbf{x})), \\
& \text{subject to} && g_j(\mathbf{x}) \leq 0, \quad j = 1, \dots, J, \\
& && h_k(\mathbf{x}) = 0, \quad k = 1, \dots, K, \\
& && x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, \dots, n.
\end{aligned} \tag{MOO}$$

Here, n and M denote the number of variables and objectives, respectively, whereas J and K denote the numbers of inequality and equality constraints, respectively. The set of all feasible solutions satisfying variable bounds and constraints is denoted as \mathbf{S} . Solving (MOO) yields a set of N Pareto-optimal (PO) solutions $\mathbf{x}^{*(o)}$, $o = 1, \dots, N$. Generally, iMCDM procedures uses a parameterized function $s(\mathbf{x}, \mathbf{p})$ provided in Equation (1) to compute a single preferred solution:

$$\mathbf{x}^{\text{pref}} = \underset{o=1, \dots, N}{\operatorname{argmin}} s(\mathbf{x}^{*(o)}, \mathbf{p}). \tag{1}$$

Achievement Scalarizing Function (ASF) [18] is a common choice for $s(\cdot)$. Where $\mathbf{p} = (\bar{\mathbf{z}}, \mathbf{w})$, is preference information in terms of reference point ($\bar{\mathbf{z}}$) and weight vector (\mathbf{w}). Unlike *a posteriori* methods, iMCDM procedures solve a parameterized and scalarized optimization problem iteratively, given the current preference information $\mathbb{P}(\mathbf{p})$, defined as

$$\begin{aligned}
& \text{Minimize} && s(\mathbf{x}, \mathbb{P}), \\
& \text{subject to} && \mathbf{x} \in \mathbf{S}.
\end{aligned} \tag{2}$$

In addition to preference information (\mathbb{P}) such as aspiration levels, weights, reference points, or objective classifications, ASF-based methods require ideal and nadir points values for normalization purpose [3].

III. PROPOSED MACHINE LEARNING-BASED DECISION-MAKER: MACHINE-DM

The outline of the proposed Machine-DM approach is presented in Figure 1. The Machine-DM approach comprises two artificial neural networks (ANN1 and ANN2) and a preference-evaluator to perform objective classification, bounding parameter prediction, and comparison of alternative PO solutions.

Following NIMBUS [15], ANN1 (\mathcal{F}_{CL}) classifies each objective $f_i^c = f_i(\mathbf{x}^{*,c})$ into one of four classes (Figure 2):

- (1) **Class $\mathbf{I}^<$** : The DM seeks improvement, i.e., $f_i(\mathbf{x}) < f_i^c$.
- (2) **Class \mathbf{I}^{\leq}** : Improvement till an aspiration level, where $\bar{z}_i < f_i^c$ and \bar{z}_i serving as a bounding parameter.
- (3) **Class $\mathbf{I}^=$** : The current value is acceptable, $f_i(\mathbf{x}) \approx f_i^c$, aiming for all objectives to fall in this class.
- (4) **Class \mathbf{I}^{\geq}** : The objective can be relaxed up to an upper bound $f_i(\mathbf{x}) \leq \varepsilon_i$ where $\varepsilon_i > f_i^c$, serving as an another bounding parameter.

A. ANN1: Class Predictor ML

Ideally, a class-predictor is an ANN model which takes a PO decision vector (\mathbf{x}^*) as input and predicts the objective classification status vector as follows:

$$\mathcal{F}_{CL} : \mathbf{x}^* \xrightarrow{\text{ANN1}} \{I^<, I^{\leq}, I^=, I^{\geq}\}^M, \quad \mathbf{x} \in \mathbb{R}^n.$$

Technically, the ANN \mathcal{F}_{CL} provides a direct mapping between a PO decision variable vector (\mathbf{x}^*) and its objective classification status (\mathbf{I}) without computing the objective vector. The advantage of using \mathcal{F}_{CL} is that it does not require any prior information on the location of the pessimistic point (R), target solution (T), and objective values. Further details on Class predictor ML is provided in Appendix A.

B. ANN2: Parameter Predictor ML

The parameter-predictor ANN model predicts the bounding parameters ($\bar{z}_i, \varepsilon_j, i \in I^{\leq}, j \in I^{\geq}$), which serve as the upper and lower bounds of the objective values for the next iteration of the iMCDM procedure. These bounds are typically supplied manually by a human DM. The parameter-predictor model replaces this step by using ANN2 (\mathcal{F}_{PP}) to directly infer the bounding parameters from a given PO decision variable vector \mathbf{x}^* , as follows:

$$\mathcal{F}_{PP} : \mathbf{x}^* \xrightarrow{\text{ANN2}} \{\bar{z}, \varepsilon, 0\}^M.$$

Here, ε is the output for each Class \mathbf{I}^{\geq} objective, \bar{z} for each Class \mathbf{I}^{\leq} objective, and zero for objectives in Classes $\mathbf{I}^<$ or $\mathbf{I}^=$. A detailed description on parameter (\bar{z}, ε) computation and associated parameters (δ, α) is provided in Appendix A.

C. Preference Evaluator

When all objectives fall into Class $\mathbf{I}^=$ (or $\mathbf{I}^= \cup \mathbf{I}^{\geq}$), the solution is considered preferred, and the iMCDM procedure can terminate. \mathcal{M}_{PE} is defined as follows:

$$\mathcal{M}_{PE} = \max_{m=1}^M |3 - I_m| + \left\{ \max(\bar{z}_i^n, \varepsilon_j^n), \forall i \in \mathbf{I}^{\leq}, \forall j \in \mathbf{I}^{\geq} \right\}, \tag{3}$$

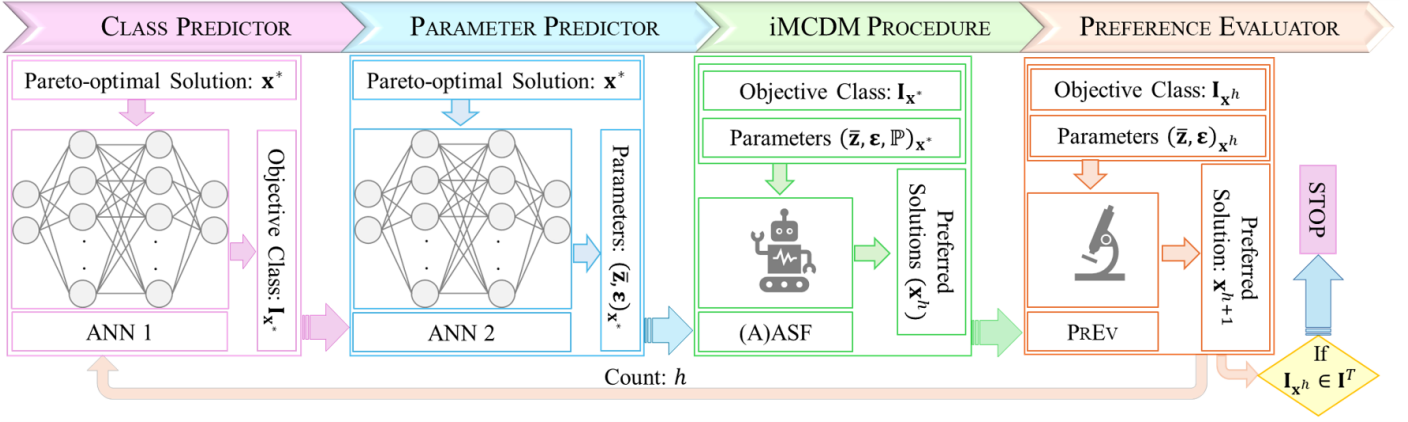


Fig. 1: The proposed approach for ML-based Machine Decision-maker (Machine-DM).

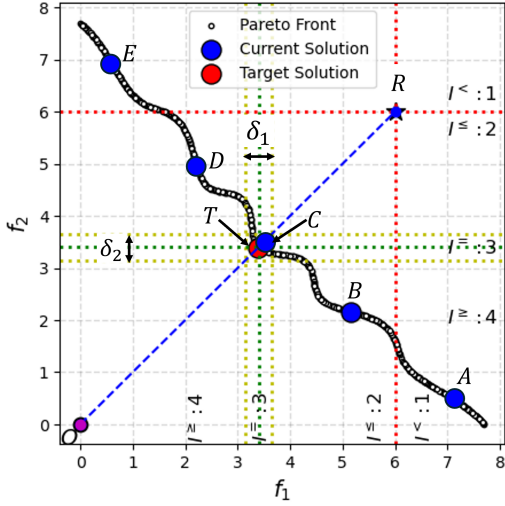


Fig. 2: Background on the Machine-DM approach. A-E are PO points located in different classes. R and O indicate a pessimistic point and ideal point, respectively. δ_1 and δ_2 are the width of Class $I^=$ and T is the target solution.

where $I_m \in \{1, 2, 3, 4\}$ for Classes $\{I^<, I^≤, I^=, I^≥\}$, and

$$\bar{z}_i^n = \frac{\bar{z}_i - f_i^T}{f_i^R - f_i^T}, \quad \varepsilon_j^n = \frac{f_j^T - \varepsilon_j}{f_j^T - f_j^*}; \quad \bar{z}_i^n, \varepsilon_j^n \in [0, 1]. \quad (4)$$

The first term measures deviation from the ideal class, while the second accounts for proximity within Classes $I^≤$ and $I^≥$. A smaller \mathcal{M}_{PE} indicates a solution closer to the target solution, enabling ranking and comparison of PO solutions.

D. Performance Evaluation Metrics

1) *Number of DM Calls*: Let h_e be the number of DM calls needed to reach a solution close to the target in experiment e . For E experiments, the DM-calls metric h_E is defined as:

$$h_E = \frac{1}{E} \sum_{e=1}^E h_e. \quad (5)$$

2) *Success Rate*: Success Rate is expressed as the proportion of the experiments in which the target class is achieved:

$$\eta_E = \frac{1}{E} \sum_{e=1}^E \{1 | I_e^T \in \{I^=, I^≥\}\}, \quad \eta_E \in [0, 1]. \quad (6)$$

3) *Deviation from Target Class*: In each independent run e , the deviation from the ideal target class ($I \in I^=$) is computed by taking the absolute difference between the each objective class and the ideal target class (Class 3) and normalized it between 0 and 1. The objective class deviation (ϕ_e) from the target class in a given experiment e is computed as:

$$\phi_e = \frac{1}{2M} \sum_{m=1}^M |3 - I_m^{T,e}|, \quad I_m^{T,e} \in \{1, 2, 3, 4\}. \quad (7)$$

$I_m^{T,e}$ is class of the final preferred solution. Finally, the metric across E experiments ($\Phi_E \in [0, 1]$) is computed as:

$$\Phi_E = \frac{1}{E} \sum_{e=1}^E \{\phi_1, \phi_2, \dots, \phi_E\}. \quad (8)$$

4) *Number of Solution Evaluations*: Number of solution evaluation across E experiments is computed, as follows:

$$N_{evl}^E = Gen \times pop \times (h_E + 1), \quad (9)$$

where Gen and pop are the number of generations and population size of real-coded GA used to solve the ASF problem. To take care of the initial feasible starting point, an additional RGA to h_E calls is needed.

IV. MACHDM-GUESS PROCEDURE

The MachDM-GUESS procedure has the following steps:

- (1) Calculate ideal objective vector (\mathbf{z}^*) and nadir objective vector (\mathbf{z}^{nad}). Set iteration $h = 0$ and DM call $h_e = 0$.
- (2) Compute a feasible start point $\mathbf{x}^{(0)}$ and corresponding objective vector $\mathbf{z}^{(0)}$. Set $h = 1$. Solve the following AASF subproblem (SP-0) to compute start point ($\mathbf{z}^{(1)}$):

$$\text{Minimize } \max_{i=1}^M \left[\frac{f_i(\mathbf{x}) - z_i^{(0)}}{z_i^{nad} - z_i^{**}} + \rho \sum_{i=1}^M \frac{f_i(\mathbf{x})}{z_i^{nad} - z_i^{**}} \right] \quad (\text{SP-0})$$

subject to $\mathbf{x} \in \mathbf{S}$ and \mathbf{z}^{**} is a utopian point.

- (3) For $\mathbf{z}^{(h)}$, predict the objective function classification ($\mathbf{I} \in \{I^<, I^{\leq}, I^=, I^{\geq}\}$) and parameters ($\varepsilon_i^h, \forall i \in I^{\leq}; \bar{z}_j^h, \forall j \in I^{\geq}$) using Class Predictor ML (\mathcal{F}_{CL}) and Parameter Predictor ML (\mathcal{F}_{PP}), respectively. This step is a Machine-DM call. Set $h_e = h_e + 1$. If all the objectives belong to class $\{I^=, I^{\geq}\}$, go to Step (7).
- (4) Use the predicted parameters ($\varepsilon_i^h, \forall i \in I^{\leq}; \bar{z}_j^h, \forall j \in I^{\geq}$) to specify upper or lower bounds to the objective functions and solve the following ASF problem (SP-I) using a reference point $\hat{\mathbf{z}}^h$ as provided in Subsection V-B.

$$\begin{aligned} & \text{Minimize} && \max_{i=1}^M \left[\frac{f_i(\mathbf{x}) - z_i^{\text{nad}}}{z_i^{\text{nad}} - \hat{z}_i^h} \right], && \text{(SP-I)} \\ & \text{subject to} && \mathbf{x} \in \mathbf{S}. \end{aligned}$$

- (5) Use Preference Evaluator to compute \mathcal{M}_{PE} values for current solution (\mathbf{x}^h) and compare it with previous solution's ($\mathbf{x}^1, \dots, \mathbf{x}^{h-1}$) \mathcal{M}_{PE} values to rank the solutions according to their ascending order of \mathcal{M}_{PE} values.
- (6) For current solution (\mathbf{x}^h), if $\mathbf{x}^h = \mathbf{x}^{h+1}$, go to step (7). Otherwise, set $h = h + 1$ and go to step (3).
- (7) Stop. The final solution is \mathbf{x}^h .

V. PROPOSED OBJECTIVE BOUND AND REFERENCE POINT COMPUTATION FOR MACHDM-GUESS PROCEDURES

The MachDM-GUESS procedure requires computation of objective bounds and generating reference point within defined objective bounds [16]. In this section, we propose systematic approaches for updating objective bounds in each iteration of MachDM-GUESS procedure and sampling random reference point $\hat{\mathbf{z}}^h$ within updated objective bounds.

A. Updating upper and lower bounds of objectives

The lower bound (\mathbf{z}^L) and upper bounds (\mathbf{z}^U) of objectives can be updated using two methods discussed as follows.

1) *Using current solution (\mathbf{z}^h) only:* For Step 1 ($h = 0$), initialize lower bound $z_i^L = z_i^*$ and upper bound $z_i^U = z_i^{\text{nad}}$. For iteration $h > 0$, update the upper and lower bounds as follows (\mathbf{z}^h is the current preferred solution): (i) $z_i^U = z_i^h$, if $z_i^h \in \{I^<, I^{\leq}\}$, (ii) $z_i^L = z_i^U = z_i^h$, if $z_i^h \in I^=$, and (iii) $z_i^L = z_i^h$, if $z_i^h \in I^{\geq}$. ANN2 is not used at all.

2) *Using current solution (\mathbf{z}^h) and parameters (\bar{z}, ε):* For iteration $h = 0$, initialize: $z_i^L = z_i^*$, $z_i^U = z_i^{\text{nad}}$. For iteration $h > 0$, consider \mathbf{z}^h is the current solution and $\bar{z}_i (\forall i \in I^{\leq}), \varepsilon_i (\forall i \in I^{\geq})$ are bounding parameters from ANN2. Update the upper and lower bound as follows: (i) $z_i^U = z_i^h$, if $z_i^h \in I^<$, (ii) $z_i^U = \bar{z}_i$, if $z_i^h \in I^{\leq}$, (iii) $z_i^L = z_i^U = z_i^h$, if $z_i^h \in I^=$, and (iv) $z_i^L = z_i^h, z_i^U = \varepsilon_i$ if $z_i^h \in I^{\geq}$.

B. Sampling reference point within objective bounds

We propose the following two approaches for sampling reference point within objective bounds, as follows.

1) *Using uniform distribution:* Consider $[z_i^L, z_i^U]$ be the bounds for objective f_i . The reference point for objective f_i is computed as $\hat{z}_i = z_i^L + (z_i^U - z_i^L) \times \mathbb{U}[0, 1]$. Where, $\mathbb{U}[0, 1]$ is uniform random number between 0 and 1.

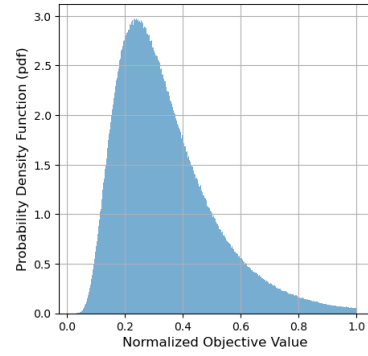


Fig. 3: Computation of reference point $\hat{\mathbf{z}}$ using log-normal distribution ($\text{LN}(\mu, \sigma)$) truncated between $[0, 1]$, mapped to the current objective range of the bounding box.

2) *Using truncated log-normal distribution:* A uniform distribution may produce a stochastic reference point far from the current solution, and may produce instability among subsequent iterations, which may be far from a human DM's perception. Hence, we propose a reference point sampling technique which randomly generates a reference point close to the current solution following a log-normal distribution, as presented in Figure 3. Given the objective bounds $[z_i^L, z_i^U]$, the reference point for objective f_i is computed as:

$$\begin{aligned} \hat{z}_i &= z_i^L + (z_i^U - z_i^L) \times \text{LN}(\mu, \sigma), \quad \forall i \in I^{\geq}, \\ \hat{z}_i &= z_i^U - (z_i^U - z_i^L) \times \text{LN}(\mu, \sigma), \quad \forall i \in \{I^<, I^{\leq}\}. \end{aligned}$$

Note that for objectives $f_i \in I^=$, $z_i^L = z_i^U$. Here, μ and σ is the mean and variance of the truncated LN distribution.

C. MachDM-GUESS Procedures

In the following, we outline the three variants of MachDM-GUESS procedures based on the choice of the proposed objective bound computation and reference point generation.

1) *Original MachDM-GUESS Procedure:* The original MachDM-GUESS computes objective bounds using the current solutions as well as bounding parameters. However, this procedure uses the uniform distribution for generating a reference point within objective bounds.

2) *Modified MachDM-GUESS-I Procedure:* The proposed MachDM-GUESS-I computes objective bounds using the current solution only. Whereas, for generating a reference point within objective bound, MachDM-GUESS-I procedure selects a point using the proposed log-normal distribution.

3) *Modified MachDM-GUESS-II Procedure:* The proposed MachDM-GUESS-II computes objective bounds using the current solution and bounding parameters. However, for generating a reference point within objective bound, MachDM-GUESS-II procedure selects a point using the proposed log-normal distribution.

VI. SIMULATION AND RESULTS

We demonstrate the comparison of three GUESS variants—original GUESS and two modified GUESS procedures—on two to eight-objective problems with up to 10 variables and

TABLE I: Problem description and performance comparison of exiting and modified GUESS procedures following Machine-DM approach. M , n , and J denotes number of objective, variables, and constraints, respectively. The terms h , η , Φ , and N_{eval}^E are performance metrics computed on 11 independent runs. Performance metrics are defined in Section III-D.

Problems	Problem Description			NSGA-II/III Parameters		MachDM-GUESS Procedure $\hat{\mathbf{z}} \sim \mathcal{U} \wedge (\bar{\mathbf{z}}, \varepsilon, \mathbf{z}^h)$ -bound				MachDM-GUESS-I Procedure $\hat{\mathbf{z}} \sim \mathcal{LN} \wedge \mathbf{z}^h$ -bound				MachDM-GUESS-II Procedure $\hat{\mathbf{z}} \sim \mathcal{LN} \wedge (\bar{\mathbf{z}}, \varepsilon, \mathbf{z}^h)$ -bound			
	M	N	J	pop	Gen	h_E	η_E	Φ_E	N_{eval}^E	h_E	η_E	Φ_E	N_{eval}^E	h_E	η_E	Φ_E	N_{eval}^E
ZDT3	2	5	0	1000	100	3.82	0.91	0.05	9.64e+04	3.18	1.00	0.00	8.36e+04	3.73	1.00	0.00	9.45e+04
Knee	2	5	0	1000	100	4.18	1.00	0.14	1.04e+05	3.36	1.00	0.14	8.73e+04	3.82	1.00	0.14	9.64e+04
Knee	3	5	0	1350	100	8.73	0.55	0.36	1.95e+05	5.91	1.00	0.11	1.38e+05	6.27	1.00	0.12	1.45e+05
Crash.	3	5	0	1350	100	2.82	1.00	0.00	7.64e+04	2.45	1.00	0.00	6.91e+04	2.64	1.00	0.00	7.27e+04
Car Side	3	7	10	1350	100	5.00	1.00	0.08	1.20e+05	3.91	1.00	0.06	9.82e+04	4.73	1.00	0.06	1.15e+05
C2-DTLZ2	3	10	1	1350	100	5.00	1.00	0.02	1.20e+05	4.64	1.00	0.05	1.13e+05	5.09	1.00	0.00	1.22e+05
DTLZ7	3	10	0	5151	100	7.36	0.82	0.18	1.67e+05	8.64	0.55	0.27	1.93e+05	9.73	0.45	0.26	2.15e+05
River	4	2	0	2000	200	9.18	1.00	0.02	2.04e+05	8.45	0.82	0.16	1.89e+05	9.27	0.73	0.18	2.05e+05
WATER	5	3	7	2000	200	5.64	1.00	0.05	1.33e+05	5.36	0.91	0.13	1.27e+05	4.91	1.00	0.07	1.18e+05
DTLZ2	8	10	0	2000	500	12.00	0.64	0.05	2.60e+05	8.73	0.73	0.13	1.95e+05	11.73	0.45	0.14	2.55e+05
Wilcoxon Signed-rank Test Results (+/=-)						1/5/4	3/7/0	4/5/1	1/5/4	8/2/0	5/5/0	4/4/2	8/2/0	1/6/3	1/8/1	1/6/3	1/6/3
Rank based on number of + and =						3	1	1	3	1	1	2	1	2	3	2	2

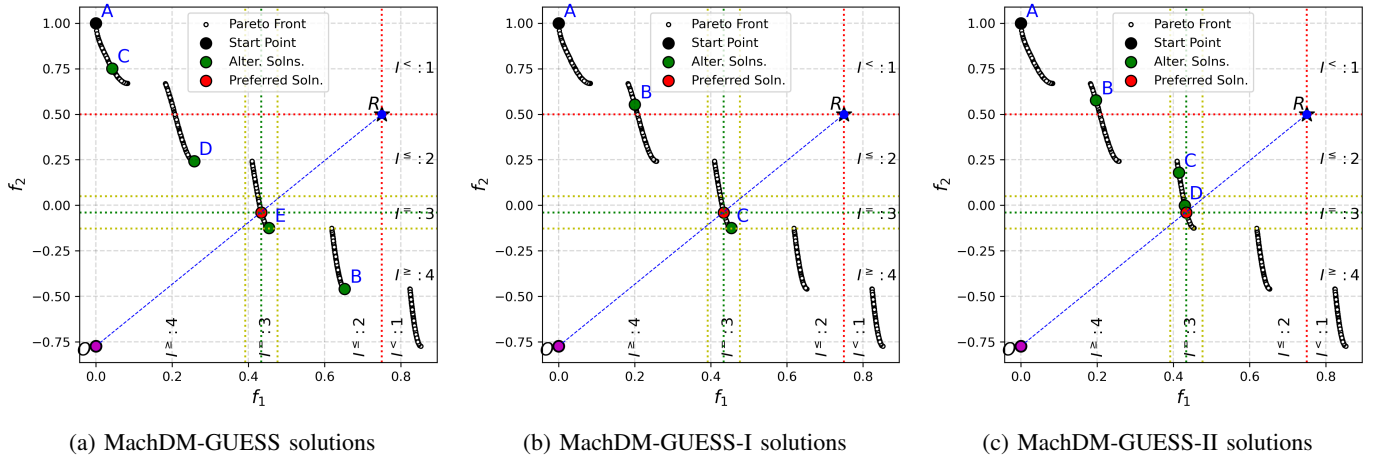


Fig. 4: Comparison of MachDM-based GUESS approach on bi-objective ZDT3 example using same start point ($\mathbf{x}^{(0)}$).

10 constraints. The description of the problems, population size and generation for computing PF using NSGA-II/NSGA-III is provided in Table I. The PF solutions \mathbf{x} and \mathbf{f} are generated using *pymoo* [19], and are used to generate the data to train ANN1 and ANN2. The augmentation parameter (ρ) for AASF formulation is set to 0.001. Subproblems (SP-0) and (SP-1) are solved using a real-coded GA (RGA) [20]. The RGA parameters –population size and number of generations– are set to 100 and 200, respectively, for each problem. Other RGA parameters are selected according to [4]. Maximum DM calls is set as $h_E^{max} = 11$ for all problems, except for eight-objective DTLZ2, for which $h_E^{max} = 15$ is chosen. The log-normal parameters $\mu = 0.35$ and $\sigma = 0.035$ are selected for two to five-objective problems. For eight-objective DTLZ2 problem, $\mu = 1.0$ and $\sigma = 1.25$ are selected. Table I present the statistical comparison of modified and original.

A. Bi-objective problems

We use ZDT3 [21] and knee problem [22]. The results for a single (out of 11) instance for ZDT3 problem using MachDM-

GUESS, MachDM-GUESS-I, and MachDM-GUESS-II procedures are provided in Figure 4 and Table II. The statistical comparison using Wilcoxon signed-rank test [23] (Table I) suggests that for bi-objective examples, MachDM-GUESS-I is the best procedure and MachDM-GUESS-II is statistically similar on all performance metrics. Also, η^E and Φ^E metrics for MachDM-GUESS is statistically similar.

B. Three-objective problems

Three-objective examples include knee [22], vehicle crashworthiness [24], car side impact [25], C2-DTLZ2 [5] and DTLZ7 [26] problems. Table I highlights that MachDM-GUESS-I procedure has best performance for all the three-objective cases except for DTLZ7. For DTLZ7 MachDM-GUESS performs the best and MachDM-GUESS-II remains statistically similar to MachDM-GUESS.

C. Many-objective problems

Many objective examples include four-objective river pollution [24], five-objective WATER [4], and eight-objective

TABLE II: Iteration-wise solutions obtained in MachDM-GUESS procedures for two-objective ZDT3 with a start point $\mathbf{x}^{(0)}=[0.1570, 0.3087, 0.6443, 0.2239, 0.5238]$.

MachDM-GUESS procedure								
Points	f_1	f_2	I_1	I_2	\bar{z}_1/ε_1	\bar{z}_2/ε_2	\mathcal{M}_{PE}	Rank
A: $\mathbf{z}^{(1)}$	0.0000	0.9998	4	1	0.1051	–	2.9399	5
B: $\mathbf{z}^{(2)}$	0.6525	-0.4583	2	4	0.5490	-0.1712	1.3283	3
C: $\mathbf{z}^{(3)}$	0.0429	0.7510	4	1	0.1253	–	2.8820	4
D: $\mathbf{z}^{(4)}$	0.2578	0.2422	4	2	0.3354	0.1597	1.2816	2
E: $\mathbf{z}^{(5)}$	0.4539	-0.1242	3	3	–	–	0.0236	1

MachDM-GUESS-I procedure								
Points	f_1	f_2	I_1	I_2	\bar{z}_1/ε_1	\bar{z}_2/ε_2	\mathcal{M}_{PE}	Rank
A: $\mathbf{z}^{(1)}$	0.0000	0.9998	4	1	0.1051	–	2.9399	3
B: $\mathbf{z}^{(2)}$	0.1999	0.5539	4	1	1.1261	–	2.3772	2
C: $\mathbf{z}^{(3)}$	0.4539	-0.1242	3	3	–	–	0.0236	1

MachDM-GUESS-II procedure								
Points	f_1	f_2	I_1	I_2	\bar{z}_1/ε_1	\bar{z}_2/ε_2	\mathcal{M}_{PE}	Rank
A: $\mathbf{z}^{(1)}$	0.0000	0.9998	4	1	0.1051	–	2.9399	4
B: $\mathbf{z}^{(2)}$	0.1967	0.5761	4	1	0.3979	–	2.1031	3
C: $\mathbf{z}^{(3)}$	0.4143	0.1790	3	2	–	-0.4321	1.4451	2
D: $\mathbf{z}^{(4)}$	0.4298	-0.0014	3	3	–	–	0.4113	1

DTLZ2 [26] problem. Table I infers that MachDM-GUESS-I has the best values for DM calls (h_E) and the number of solution evaluations (N_{eval}^E) for river and DTLZ2 problems. For WATER example, MachDM-GUESS-II performs the best. However, in terms of success rate (η_E) and average deviation from the target class (Φ_E), MachDM-GUESS outperforms both the modified GUESS procedures.

The overall ranking displayed in the final row highlights that the modified MachDM-GUESS-I has the best performance, followed by MachDM-GUESS. Creating $\hat{\mathbf{z}}^h$ close to the current solution \mathbf{z}^h using log-normal distribution does not require bounding parameters, thereby eliminating the need of ANN2.

VII. CONCLUSION

This paper has proposed two variants of the GUESS MCDM procedure – GUESS-I and GUESS-II. For the performance evaluation of GUESS-I and GUESS-II procedures, a ML-based DM (Machine-DM) is trained using at most two ANNs, which replace human decisions stated in the steps of GUESS. Following Machine-DM, MachDM-GUESS, MachDM-GUESS-I and MachDM-GUESS-II have been compared using four proposed performance evaluation metrics (h_E, η_E, Φ_E , and N_{eval}^E). The statistical comparison on 11 independent runs for each problem using Wilcoxon signed-rank test [23] reveals that MachDM-GUESS-I is the overall best-performing procedure among all the three procedures.

In future, we plan to develop a comprehensive framework for benchmarking iMCDM procedures using Machine-DM. This study should lead a way and motivate EMO and MCDM researchers to develop more such computational iMCDM procedures using machine learning based DMs.

REFERENCES

[1] K. Deb, *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: Wiley, 2001.

[2] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, ser. Genetic and Evolutionary Computation. New York, NY: Springer, 2007.

[3] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[5] K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints,” *IEEE transactions on evolutionary computation*, vol. 18, no. 4, pp. 577–601, 2013.

[6] Q. Zhang and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[7] K. Li, R. Chen, G. Fu, and X. Yao, “Two-archive evolutionary algorithm for constrained multiobjective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 303–315, 2018.

[8] A. Wierzbicki, “Reference point approaches,” in *Multicriteria Decision Making*. Springer, 1999, pp. 237–275.

[9] K. Miettinen and M. M. Mäkelä, “On scalarizing functions in multiobjective optimization,” *OR spectrum*, vol. 24, no. 2, pp. 193–213, 2002.

[10] D. Yadav, P. Ramu, and K. Deb, “Visualization-aided multi-criterion decision-making using reference direction based pareto race,” in *2022 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2022, pp. 125–132.

[11] —, “Interpretable self-organizing map assisted interactive multicriteria decision-making following Pareto-Race,” *Applied Soft Computing*, vol. 149, p. 111032, 2023.

[12] R. Benayoun, J. De Montgolfier, J. Tergny, and O. Laritchev, “Linear programming with multiple objective functions: Step method (STEM),” *Mathematical programming*, vol. 1, no. 1, pp. 366–375, 1971.

[13] H. Nakayama and Y. Sawaragi, “Satisficing trade-off method for multiobjective programming,” in *Interactive Decision Analysis*. Springer, 1984, pp. 113–122.

[14] P. Korhonen and J. Wallenius, “A multiple objective LP decision support system,” *Decision Support System*, vol. 6, no. 3, pp. 243–251, 1990.

[15] K. Miettinen and M. M. Mäkelä, “NIMBUS — Interactive Method for Nondifferentiable Multiobjective Optimization Problems,” in *Multi-Objective Programming and Goal Programming*. Springer Berlin Heidelberg, 1996, pp. 50–57.

[16] J. T. Buchanan, “A naive approach for solving MCDM problems: The GUESS method,” *Journal of the Operational Research Society*, vol. 48, no. 2, pp. 202–206, 1997.

[17] K. Miettinen and M. M. Mäkelä, “Comparative evaluation of some interactive reference point-based methods for multi-objective optimisation,” *J. of the Op. Research Society*, vol. 50, no. 9, pp. 949–959, 1999.

[18] A. P. Wierzbicki, “The use of reference objectives in multiobjective optimization,” in *Multiple Criteria Decision Making Theory and Application*. Springer, 1980, pp. 468–486.

[19] J. Blank and K. Deb, “Pymoo: Multi-objective optimization in python,” *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[20] K. Deb, “A population-based algorithm-generator for real-parameter optimization,” *Soft Computing*, vol. 9, no. 4, pp. 236–253, 2005.

[21] E. Zitzler, K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: Empirical results,” *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[22] J. Branke, K. Deb, H. Dierolf, and M. Osswald, “Finding Knees in Multi-objective Optimization,” in *Parallel Problem Solving from Nature*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 722–731.

[23] T. Harris and J. W. Hardin, “Exact Wilcoxon signed-rank and Wilcoxon Mann–Whitney ranksum tests,” *The Stata Journal*, vol. 13, no. 2, pp. 337–343, 2013.

[24] R. Tanabe and H. Ishibuchi, “An easy-to-use real-world multi-objective optimization problem suite,” *Appl. Soft Comput.*, vol. 89, p. 106078, 2020.

[25] Y. Vesikar, K. Deb, and J. Blank, “Reference point based NSGA-III for preferred solutions,” in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 1587–1594.

[26] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable test problems for evolutionary multiobjective optimization,” in *Evolutionary multiobjective optimization: theoretical advances and applications*. Springer, 2005, pp. 105–145.

APPENDIX A

DETAILS ON CLASS PREDICTOR ML: ANN1

Ideally, the class-predictor illustrated in Figure 5 is an ANN that takes a decision vector (\mathbf{x}) as input and outputs the corresponding objective classification status vector:

$$\mathcal{F}_{CL} : \mathbf{x} \xrightarrow{\mathcal{F}_{CL}} \{I^<, I^{\leq}, I^=, I^{\geq}\}^M, \quad \mathbf{x} \in \mathbb{R}^n.$$

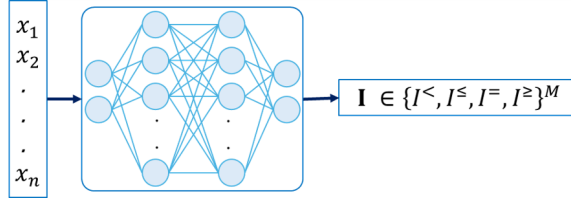


Fig. 5: Class predictor ML (\mathcal{F}_{CL}): $\mathbf{x}(\in \mathbb{R}^n) \rightarrow \mathbf{I}(\in \mathbb{Z}^M)$. $\mathbb{Z} = \{1, 2, 3, 4\}$ represents four Classes $\{I^<, I^{\leq}, I^=, I^{\geq}\}$.

In essence, \mathcal{F}_{CL} establishes a direct mapping from a decision vector (\mathbf{x}) to its objective classification status (\mathbf{I}) without explicitly computing the objective values. A key advantage of \mathcal{F}_{CL} is that it requires no prior knowledge of the pessimistic point (R), the target solution (T), or the actual objective values (refer to Figure 2). Consequently, using this ANN for a given \mathbf{x} eliminates the need for costly objective function evaluations. In this sense, ANN1 serves both as a surrogate model and as a classifier, providing the objective classification.

DETAILS ON PARAMETER PREDICTOR ML: ANN2

The parameter-predictor ANN (Figure 6) estimates the bounding parameters ($\bar{z}_i, \varepsilon_j; i \in I^{\leq}, j \in I^{\geq}$), which define the lower and upper bounds of objective values for the subsequent iteration of the iMCDM procedure. Traditionally, these bounds are provided manually by a human decision-maker. The parameter-predictor ANN (\mathcal{F}_{PP}) automates this process by directly inferring the bounding parameters from a given decision vector \mathbf{x} :

$$\mathcal{F}_{PP} : \mathbf{x} \xrightarrow{\mathcal{F}_{PP}} \{\bar{z}, \varepsilon, 0\}^M.$$

Here, ε_j corresponds to each Class I^{\geq} objective, \bar{z}_i corresponds to each Class I^{\leq} objective, and zero is assigned for objectives in Classes $I^<$ or $I^=$. Specifically, ε_j represents the upper bound for the j -th Class I^{\geq} objective (as identified by

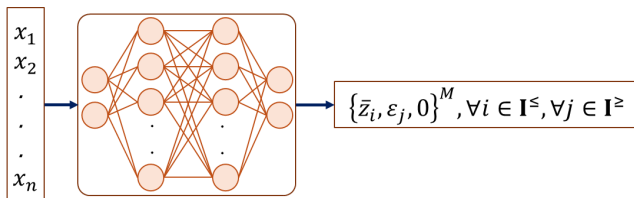


Fig. 6: Parameter predictor ML (\mathcal{F}_{PP}): $\mathbf{x} \rightarrow \{\bar{z}, \varepsilon, 0\}^M$.

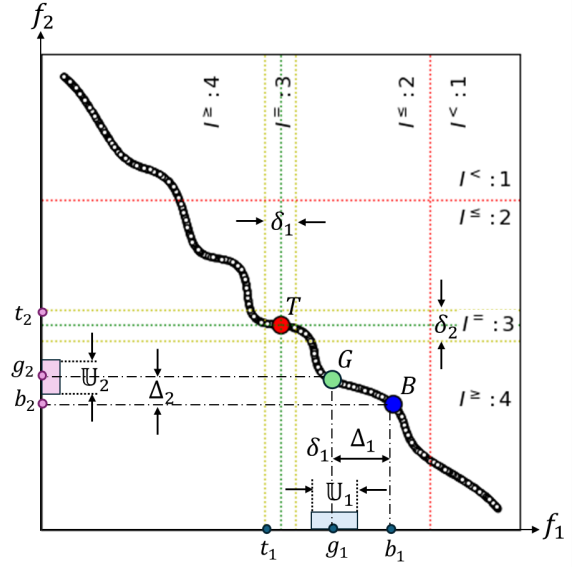


Fig. 7: Defining the bounding parameters (\bar{z}_i, ε_j) for parameter-predictor ML.

\mathcal{F}_{CL}), ensuring $f_j(\mathbf{x}) \leq \varepsilon_j$, while \bar{z}_i is the lower bound for the i -th Class I^{\leq} objective, requiring $f_i(\mathbf{x}) \geq \bar{z}_i$. For objectives in Classes $I^<$ and $I^=$, no bounds are necessary, so \mathcal{F}_{PP} outputs zero.

For a Class I^{\geq} objective, the upper bound ε_j is determined based on the current objective value $f_j^c = f_j(\mathbf{x}^{*,c})$ and the perceived target f_j^T . When f_j^c is far from f_j^T , the decision-maker may be uncertain about the desired improvement. This uncertainty is addressed by computing Δ_j as shown in Figure 7 and defined in Equation (10):

$$\Delta_j = \min [\alpha_j (z_j^{\text{nad}} - z_j^*), (f_j^T - f_j^c) + 0.5\delta_j], \quad (10)$$

where α_j is a problem-specific parameter, and δ_j defines the Class $I^=$ range, as illustrated in Figure 7. The upper bound ε_j is then set with a small uniform randomness (\mathbb{U}) to reflect human uncertainty:

$$\varepsilon_j = f_j^c + (1 + \alpha_j \mathbb{U}[-1, 1])\Delta_j, \quad (11)$$

ensuring that $\varepsilon_j > f_j^c$ when $\alpha_j < 1$. Similarly, for Class I^{\leq} objectives, the lower bound \bar{z}_i is computed as:

$$\bar{z}_i = f_i^c - (1 + \alpha_i \mathbb{U}[-1, 1])\Delta_i, \quad (12)$$

with Δ_i defined analogously to Equation (10) using index i . This ensures $\bar{z}_i < f_i^c$ for $\alpha_i < 1$. The inclusion of \mathbb{U} introduces variability in the bounds, mimicking the inexact responses of a human DM.

APPENDIX B

DATA GENERATION FOR TRAINING ANNS

In this section, we provide a comprehensive computational framework for generating data for training ANN1 and ANN2 in detail. For a given M -objective and n -variable problem, generate a well distributed, well-converged, and a diverse set of near Pareto optimal solutions ($[\mathbf{x}^*, \mathbf{f}^*]_{i=1}^N$). Consider that

TABLE III: Classification status for two-objective PF solutions presented in Figure 2.

Class I	f_1 Range	f_2 Range	Representative PO-Point
$[I^<, I^>]$	$> f_1^R$	$< f_2^T - 0.5\delta_2$	A
$[I^{\leq}, I^{\geq}]$	$[f_1^T + 0.5\delta_1, f_1^R]$	$< f_2^T - 0.5\delta_2$	B
$[I^=, I^=]$	$[f_1^T - 0.5\delta_1, f_1^T + 0.5\delta_1]$	$[f_2^T - 0.5\delta_2, f_2^T + 0.5\delta_2]$	C
$[I^{\geq}, I^{\leq}]$	$< f_1^T - 0.5\delta_1$	$[f_2^T + 0.5\delta_2, f_2^R]$	D
$[I^{\geq}, I^<]$	$< f_1^T - 0.5\delta_1$	$> f_2^R$	E

TABLE IV: Problem description and parameter setting for GA and Machine-DM. Parameters M , n , and J are number of objectives, variables, and constraints, respectively.

Examples	Problem Details			Machine-DM Parameters		ANN1/ANN2 Parameters		
	M	n	J	$\hat{\delta}$	$\hat{\alpha}$	lr_1	lr_2	l_{reg}
ZDT3	2	5	0	0.100	0.100	0.0100	0.010	0.010
Knee	2	5	0	0.100	0.075	0.0100	0.010	0.010
Knee	3	5	0	1	0.050	0.100	0.0010	0.001
Crash.	3	5	0	0.100	0.050	0.0010	0.002	0.020
Car Side	3	7	10	0.100	0.075	0.0020	0.001	0.012
C2-DTLZ2	3	10	1	0.100	0.100	0.0001	0.0001	0.001
DTLZ7	3	10	0	0.100	0.075	0.001	0.001	0.001
River	4	2	0	0.015	0.025	0.0025	0.001	0.010
WATER	5	3	7	0.025	0.035	0.0010	0.002	0.020
DTLZ2	8	10	0	0.050	0.200	0.0010	0.001	0.010

$R = \mathbf{f}^R$ is an unknown M -dimensional pessimistic point and $T = \mathbf{f}^T$ is the M -dimensional target point. Machine-DM parameters described in main manuscript are δ and α . Also, \mathbf{z}^* and \mathbf{z}^{nad} represent ideal and nadir point, respectively.

A. Data Generation for ANN1

For a M -dimensional PO point \mathbf{f}^* , the M -dimensional objective class \mathbf{I} is computed as follows:

$$I_m = \begin{cases} 1 : I^<, & \text{if } f_m^* > f_m^R \\ 2 : I^{\leq}, & \text{if } f_m^R \geq f_m^* > f_m^T + 0.5 \times \hat{\delta}_m \\ 3 : I^=, & \text{if } f_m^T - 0.5 \times \hat{\delta}_m \geq f_m^* \geq f_m^T + 0.5 \times \hat{\delta}_m \\ 4 : I^{\geq}, & \text{if } f_m^* < f_m^T - 0.5 \times \hat{\delta}_m \end{cases} \quad (13)$$

$m = 1, \dots, M.$

Here, $\hat{\delta}_m = \delta_m \times (z_m^{nad} - z_m^*)$ defines the width of target objective class ($I^=$) in m -th objective. Next, for all the PO solutions $([\mathbf{x}^*, \mathbf{f}^*]_{i=1}^N)$ obtain the class according to Equation (13). Get the training data (\mathcal{D}_{ANN1}) for ANN1 as follows:

$$\mathcal{D}_{ANN1} = [\mathbf{x}^{*,(i)}, \mathbf{I}^{(i)}]_{i=1}^N \quad (14)$$

Next, the data \mathcal{D}_{ANN1} is used for train-test-validation split and for training ANN1 (\mathcal{F}_{CL}).

B. Data Generation for ANN2

In this section, we provide the computational framework for generating training data for ANN2 (\mathcal{F}_{PP}). Following the

objective classification definition provided in Equation (13), compute the M -dimensional bounding parameter (Λ) as follows:

$$\Lambda_m = \begin{cases} 0 & \text{if } I_m \in \mathbf{I}^< \\ \bar{z}_m, & \text{if } I_m \in \mathbf{I}^{\leq} \\ 0 & \text{if } I_m \in \mathbf{I}^= \\ \varepsilon_m, & \text{if } I_m \in \mathbf{I}^> \end{cases}; \quad m = 1, \dots, M. \quad (15)$$

The detailed computation for \bar{z} and ε is provided in Section A. Next, for all the PO solutions $([\mathbf{x}^*, \mathbf{f}^*]_{i=1}^N)$, compute the bounding parameter vector (Λ). Get the training data (\mathcal{D}_{ANN2}) as follows:

$$\mathcal{D}_{ANN2} = [\mathbf{x}^{*,(i)}, \Lambda^{(i)}]_{i=1}^N. \quad (16)$$

Next, the data \mathcal{D}_{ANN2} is used for train-test-validation split and for training ANN2 (\mathcal{F}_{PP}).

APPENDIX C

ARCHITECTURE AND ERROR METRIC FOR ANNS

The architecture of ANN1 and ANN2 is provided in Table VI and Table VII, respectively. For ANN1 and ANN2, four-layered feed-forward neural networks are used with a reduced number of hidden layers [128, 64, 32, 16] and *Adam* optimizer [27] with a learning rate set to lr_1 and lr_2 , respectively. ANN1 uses *Softmax* and ANN2 uses *Sigmoid* activation functions. For ANN2, L2-regularization with parameter l_{reg} is used to train the respective ANN. For both the ANNs, 80% of data is used in training and 20% data is used for testing. In training phase, 20% of training data is used for cross-validation purpose. The batch-size and maximum epoch for both the ANNs are set to 32 and 500, respectively. For training ANN1, error metrics – Accuracy, Precision, Recall, and F-1 Score – are checked for their closeness to one. For training ANN2, R^2 and MSE is used as error metric.

CONCLUSIONS

This paper has proposed two efficient variants of GUESS method– GUESS-I and GUESS-II. Using Machine-DM, three Machine-DM based procedures–MachGUESS (original), MachDM-GUESS-I (modified), and MachDM-GUESS-II (modified) has been developed. Then these three MachDM-iMCDM procedures have been statistically compared using four newly introduced performance metrics using Wilcoxon Signed-rank Test [23]. Implementation on six benchmark and four real-world example reveals that modified GUESS-I method is the most efficient among three approaches.

TABLE V: Error metric for ANN models across benchmark MCDM problems and instances.

S.I.	Example	Objective function	ANN1 (\mathcal{F}_{CL}) Error Metric				ANN2 (\mathcal{F}_{PP}) Error Metric	
			Accuracy	Precision	Recall	F1-Score	R^2	MSE
1	ZDT3 (bi-objective)	f_1	1.0000	1.0000	1.0000	1.0000	0.9962	0.0002
		f_2	0.9429	0.9167	0.9474	0.9206	0.9566	0.0056
2	Knee (two-objective)	f_1	1.0000	1.0000	1.0000	1.0000	0.9956	0.0134
		f_2	1.0000	1.0000	1.0000	1.0000	0.9965	0.0108
3	Knee (three-objective)	f_1	0.9554	0.9482	0.9296	0.9372	0.9930	0.0198
		f_2	0.9732	0.9516	0.9609	0.9550	0.9953	0.0133
		f_3	0.9286	0.8947	0.9023	0.8983	0.9780	0.0321
4	Vehicle Crashworthiness (three-objective)	f_1	0.9610	0.9495	0.9518	0.9494	0.9962	0.0026
		f_2	0.9610	0.9651	0.9453	0.9522	0.9915	0.0066
		f_3	1.0000	1.0000	1.0000	1.0000	0.9975	0.0023
5	Car side Impact (three-objective)	f_1	0.9505	0.9259	0.9489	0.9364	0.9952	0.0049
		f_2	0.9286	0.9059	0.8633	0.8752	0.9973	0.0028
		f_3	0.9670	0.9468	0.8982	0.9201	0.9950	0.0042
6	C2-DTLZ2 (three-objective)	f_1	0.9882	0.9696	0.9622	0.9658	0.9898	0.0253
		f_2	1.0000	1.0000	1.0000	1.0000	0.9889	0.01265
		f_3	0.9704	0.9464	0.9511	0.9450	0.9927	0.0219
7	DTLZ7 (three-objective)	f_1	0.9908	0.9837	0.9659	0.9933	0.9966	0.0002
		f_2	0.9662	0.9656	0.9380	0.9512	0.9959	0.0003
		f_3	0.9415	0.8233	0.7799	0.7985	0.9863	0.0049
8	River Pollution (four-objective)	f_1	0.9950	0.9947	0.9976	0.9961	0.9987	0.0003
		f_2	0.9750	0.9421	0.9647	0.9514	0.9939	0.0001
		f_3	0.9600	0.9681	0.9059	0.9447	0.9954	0.0196
		f_4	0.9650	0.9748	0.9359	0.9527	0.9931	0.0535
9	WATER (five-objective)	f_1	0.9600	0.8344	0.9132	0.8605	0.9992	0.0008
		f_2	0.9475	0.8056	0.8211	0.8130	0.9996	0.0004
		f_3	0.9800	0.7417	0.7367	0.7391	0.9997	0.0003
		f_4	0.9775	0.7606	0.8051	0.7163	0.9994	0.0004
		f_5	0.9425	0.7073	0.7137	0.7013	0.9988	0.0012
10	DTLZ2 (eight-objective)	f_1	0.9503	0.9057	0.9121	0.9080	0.9830	0.0478
		f_2	0.9526	0.9184	0.9159	0.9161	0.9756	0.0625
		f_3	0.9464	0.9364	0.8994	0.9165	0.9786	0.0601
		f_4	0.9619	0.9382	0.9199	0.9278	0.9786	0.0677
		f_5	0.9611	0.9513	0.9603	0.9553	0.9882	0.0383
		f_6	0.9580	0.9358	0.9244	0.9299	0.9860	0.0438
		f_7	0.9565	0.9430	0.9249	0.9319	0.9845	0.0581
		f_8	0.9775	0.9727	0.9743	0.9734	0.9926	0.0222

 TABLE VI: A generic architecture of ANN1–Class Predictor ANN (\mathcal{F}_{CL}) for an n -variable, M -objective MOO problem, trained with fixed hidden (dense) layer [128,64,32,16].

Layer Type	Output Shape	Number of Parameters	Connected to
Input Layer	n	0	—
Dense Layer 1	128	$128(n+1)$	Input Layer
Dense Layer 2	64	8,256	Dense Layer 1
Dense Layer 3	32	2,080	Dense Layer 2
Dense Layer 4	16	528	Dense Layer 3
Output Layer 1	1	68	Dense Layer 4
\vdots	\vdots	\vdots	\vdots
Output Layer M	1	68	Dense Layer 4
Total Trainable Parameters:		$128n + 10,992 + 68M$	

 TABLE VII: A generic architecture of ANN2–Parameter Predictor ANN (\mathcal{F}_{PP}) for an n -variable, M -objective MOO problem, trained with fixed hidden (dense) layer [128,64,32,16].

Layer Type	Output Shape	Number of Parameters	Connected to
Input Layer	n	0	—
Dense Layer 1	128	$128(n+1)$	Input Layer
Dense Layer 2	64	8,256	Dense Layer 1
Dense Layer 3	32	2,080	Dense Layer 2
Dense Layer 4	16	528	Dense Layer 3
Output Layer 1	1	17	Dense Layer 4
\vdots	\vdots	\vdots	\vdots
Output Layer M	1	17	Dense Layer 4
Total Trainable Parameters:		$128n + 10,992 + 17M$	