Enhanced *Innovized* Progress Operator for Evolutionary Multi- and Many-Objective Optimization

Sukrit Mittal, Dhish Kumar Saxena, Kalyanmoy Deb, *Fellow, IEEE*, and Erik D. Goodman

***Abstract*—*Innovization* is a task of learning common relationships among some or all of the Pareto-optimal (PO) solutions in multi- and many-objective optimization problems. A recent study has shown that a chronological sequence of nondominated solutions obtained along the successive generations of an optimizer possesses salient patterns that can be learnt using a Machine Learning (ML) model, and can help the offspring solutions *progress* in useful directions. This article enhances each constitutive module of the above approach, including novel interventions on management of the convergence-diversity tradeoff while mapping the solutions from the previous and current generation; use of a computationally more efficient ML method, namely, Random Forest (RF); and changing the manner and extent to which the learnt ML model is utilized toward advancement of the offspring. The proposed modules constitute what is called the *enhanced innovized progress (IP2)* operator. To investigate the search efficacy provided by the IP2 operator, it is integrated with multi- and many-objective optimization algorithms, such as NSGA-II, NSGA-III, MOEA/D, and MaOEA-IGD, and tested on a range of two- to ten-objective test problems, and five real-world problems. Since the IP2 operator utilizes the history of gradual and progressive improvements in solutions over generations, without requiring any additional solution evaluations, it opens up a new direction for ML-assisted evolutionary optimization.**

***Index Terms*—*Innovization*, *Innovized Progress* (IP), learning-assisted optimization, machine learning (ML), multiobjective optimization, online *Innovization*.**

I. Introduction

OPTIMIZATION is an iterative process of arriving at one or more optimal solution(s), depending on the number of objectives (M) involved. Unlike the case of a single-objective ($M = 1$) problem, where the aim is usually to find the unique global optimum, the aim in the case of multiobjective ($M = 2$ and 3) or many-objective ($M \geq 4$) problems is to find a set of well-distributed Pareto-optimal (PO) solutions, generally referred to, in the objective (F)-space, as the Pareto Front (PF). This could be accomplished by using point-based or population-based algorithms. The latter, however, are more suited for tackling multi- and many-objective problems, since the PF approximation can be achieved in a single-algorithmic run. Since evolutionary algorithms use a population of solutions in each iteration, their extensions—evolutionary multiobjective optimization (EMO) or evolutionary many-objective optimization (EMaO) algorithms, are extensively used [1], [2].

EMO/EMaO algorithms, including NSGA-II [3], NSGA-III [4], [5], and MOEA/D [6], create new solutions (offspring) using bio-inspired genetic operators, such as crossover and mutation. Others, such as estimation of distribution algorithms (EDAs), create offspring through sampling by use of probability models, such as Bayesian networks, decision trees, etc. EDAs, such as EDA-VNS [7] and HMOBEDA [8], have shown a potentially distinctive advantage of using the intervariable relationships in generating new offspring. This linkage-preserving advantage may get more important in problems with highly linked variables. However, there are other problems in which independent mating of variables through operators like SBX crossover causes a more efficient search [9].

Some studies in the EMO/EMaO domain have focused on an *online innovization* approach [10], [11]. Here, the focus is on extracting intervariable relationships from current nondominated solutions, and utilizing these in subsequent generations to *repair* the offspring generated using the genetic operators. This approach helps *explore* the advantage of independent variable mating and also *exploit* the learnt relationships. Recently, Mittal *et al.* [12] proposed an innovized repair (IR) operator consisting of three modules: 1) training-dataset generation by mapping the solutions from earlier generations of an EMO run to the current nondominated solutions; 2) artificial neural network (ANN) training to learn the patterns in the training dataset; and 3) IR to repair some of the offspring in the current generation, using the trained ANN. The proof of concept provided in [12] paved the way for a more exhaustive analysis in [9]. In that, the term IR is replaced by innovized progress (IP), considering that: 1) it helps some of the offspring *advance* along the directions adopted by earlier solutions to evolve to

Manuscript received 27 February 2021; revised 31 May 2021, 21 August 2021, and 26 October 2021; accepted 15 November 2021. Date of publication 1 December 2021; date of current version 3 October 2022. This work was supported by the Ministry of Human Resource Development, Government of India through the SPARC Scheme under Project P66. *(Corresponding author: Dhish Kumar Saxena.)*

Sukrit Mittal and Dhish Kumar Saxena are with the Department of Mechanical and Industrial Engineering, Indian Institute of Technology Roorkee, Roorkee 247667, India (e-mail: smittal1@me.iitr.ac.in; dhish.saxena@me.iitr.ac.in).

Kalyanmoy Deb and Erik D. Goodman are with the BEACON Center for the Study of Evolution in Action and the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: kdeb@egr.msu.edu; goodman@egr.msu.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TEVC.2021.3131952.

Digital Object Identifier 10.1109/TEVC.2021.3131952

the current set of nondominated solutions and 2) the term *repair* conventionally implies modifying an infeasible solution to a feasible one.

In this article, we propose an enhanced version of the IP operator, referred to as the *enhanced IP (IP2)* operator. While the basic endeavor remains intact—namely, to capture the directional improvements in the decision space (X) to help the offspring advance more effectively, the manner in which this is accomplished, differs. In IP2, each constitutive module of the IP operator is modified or enhanced, including: 1) training-dataset generation in such a manner that the convergence-diversity tradeoff is better dealt with; 2) training a machine learning (ML) model, that is, computationally more efficient; and 3) changing the manner in which original variable bounds are honored after application of the ML model, and the extent to which the ML model is exploited for advancement of the offspring. Notably, while the training-dataset generation and utilization of the trained ML model are rigidly customized for both IP and IP2 operators, there is flexibility in the choice of the underlying ML method. Either operator can use any ML method. Given the choice of ANN in the case of IP [9], the operator is referred to as IP-ANN. In this article, RF [13] has been chosen owing to its computational efficiency over ANN [14], hence, it will be called IP2-RF.

To investigate how the search efficacy of an EMO/EMaO algorithm could be impacted by integration of the proposed IP2-RF operator, we address the following.

1) It has been integrated with NSGA-II, NSGA-III, MOEA/D, and MaOEA-IGD [15]. Such a choice implicitly promises an investigation over a *dominance*-based algorithm in NSGA-II, a *decomposition*-based algorithm in MOEA/D, an *indicator*-based algorithm in MaOEA-IGD, and a hybrid algorithm in NSGA-III that relies on *dominance* and *decomposition*.
2) Since the aim is not to compare different EMO/EMaO algorithms but rather how the performance of each is impacted by integration with the IP2-RF operator, the following comparisons are presented: NSGA-II versus NSGA-II-IP2-RF, NSGA-III versus NSGA-III-IP2-RF, MOEA/D versus MOEA/D-IP2-RF, and MaOEA-IGD versus MaOEA-IGD-IP2-RF.
3) The ZDT [16], DTLZ [17], WFG [18], and MaF [19] test suites have been used, with M ranging from 2 to 10. Some of these problems manifest difficult-to-handle characteristics, such as multimodality and bias. In addition, two problems with linked variables (*L1* and *L2* [9]) are also included to test the diverse scope of application of the proposed operator. Further, five real-world problems (including one with discrete variables) have also been examined.

While the above contextualize the core of the experimental investigations in this article, these are preceded by limited experiments (involving $M = 2$ and 3) that aim to highlight the difference in the search efficacy owing to the change in the following.

1) The ML (ANN versus RF) method alone, for which the same progress operator (IP) is retained, and the following comparisons are done: NSGA-II-IP-ANN versus NSGA-II-IP-RF; NSGA-III-IP-ANN versus NSGA-III-IP-RF; and MOEA/D-IP-ANN versus MOEA/D-IP-RF.
2) The progress operator (IP versus IP2) alone, for which the same ML method (RF) is retained, and the following comparisons are done: NSGA-II-IP-RF versus NSGA-II-IP2-RF; NSGA-III-IP-RF versus NSGA-III-IP2-RF; and MOEA/D-IP-RF versus MOEA/D-IP2-RF.

The IP2 operator can be useful for solving single-objective problems as well, since it learns the directional improvements in X-space. An introductory study on using the existing IP operator for single-objective problems has been performed recently [20]. However, the scope of this article is restricted to only multi- and many-objective optimization. The remainder of this article is structured as summarized as follows. We begin by providing an overview of the existing learning-assisted EMO/EMaO algorithms, including the state-of-the-art for *online innovization* in Section II-A, followed by coverage of the past ANN-assisted repair/progress operator in Section II-B. Section III discusses the enhancements to the IP operator, leading up to the IP2 operator. The differences between the IP and IP2 operators are highlighted in Section IV. In this background, experimental settings used in this article are presented in Section V, followed by the experimental results on a range of multi- and many-objective problems in Section VI. Some interesting insights into the efficacy of the IP2 operator are shared in Section VII, alongside consideration of five real-world problems. The potential challenges to the utility of the IP2 operator are highlighted in Section VIII, while this article concludes with Section IX.

II. Existing State of the Art

In EMO/EMaO algorithms, the genetic *variation* operators, namely, crossover and mutation, are not found effective in finding the optimal solution for certain problems [21], [22]. As a remedy, these algorithms are often coupled with other methods, including local search [23], [24] and other efficient offspring-generation (EOG) methods [25], [26]. For example, the EOG method in [25] relies on generating multiple offspring from the same set of parents (in each generation) by use of different mating strategies. This, in general, may necessitate extra solution (offspring) evaluations compared to the conventional scenario where only one/two offspring are generated from a set of parents. To limit the cost associated with extra solution (offspring) evaluations, the latter's fitness is evaluated using a surrogate model instead of the potentially expensive actual solution evaluations. This challenge of extra solution evaluation is also manifested in another EOG method [26], where besides the parents in any generation, some extra solutions which could serve as potent parents are created. Again, the evaluation of the potency/fitness of such solutions is based on surrogate models instead of the potentially expensive actual solution evaluations.

In past years, some new offspring generation methods have been proposed based on utilization of search history [27] and search directions [28]. In [27], a large number of offspring

are generated, which are then clustered based on the search history (one cluster per reference vector (RV)). Subsequently, one offspring is randomly selected from each cluster, to ensure an even distribution of the offspring across the objective (F)-space. In [28], the offspring are evolved based on some obtained search directions in the X-space. These search directions are obtained by choosing a dominated and a nondominated solution as the end points. Unlike [28], in [29], the solutions closest to the ideal point are identified as promising, and the lines joining these solutions with extremes of the search space are used as the search directions. Recently, a PCA-assisted reproduction operator [30] was proposed that builds a transformed X-space with the reduced number of dimensions. The offspring are generated in this new transformed space and then are converted back to the original X-space.

Intriguingly, the use of learning in EMO algorithms is not new. Learning, in one form or the other, has been used in EDAs [7], [8]; online innovization methods [10], [11]; surrogate-assisted optimization (SAO) algorithms [31], [32]; and EOG methods [25], [26]. Recently, some studies have shown the use of ML for achieving different goals in EMO algorithms, such as RV adaptation and mating restriction. In [33], reinforcement learning is used to adapt the RVs on-the-fly as per the needs of the problem at hand. Apart from this, a mating strategy based on manifold learning has been used to solve problems with complicated Pareto sets [34] and constraints [35].

In a significant departure from the above approaches, there have been recent studies in which the notion of *innovization* (discussed below) is extended to *learn* the pertinent directional changes in X-space (from a history of gradual and progressive improvements, accumulated over multiple generations) and utilize these to potentially improve the quality of offspring produced by conventional genetic crossover and mutation.

A. Innovization

Innovization is a task of extracting innovative design principles or relationships from the PO solutions, in the context of design variables and objectives [36], [37]. *Innovization* was initially proposed as a post-optimization process, though later studies claimed that some intervariable relationships (if they exist) can also be extracted from the nondominated solutions at an intermediate generation of an EMO run [10], [11]. These relationships can then be used to *repair* the offspring generated by crossover and mutation, so they internalize some generic properties manifested by the current nondominated solutions. This process, when repeated over multiple generations, propagates some relationships, which has been found to expedite convergence toward the PF. However, there are many choices to be made for this task to be effective, such as, how often to learn these relationships during the EMO run; how to design the learning mechanism; in which form these relationships could be learnt; how the offspring are repaired; etc.

B. ANN-Assisted Innovized Progress Operator (IP-ANN)

The study in [12] marked the first attempt to capture the direction of evolution in a multidimensional search space using an ANN and to apply it back to *repair* the offspring, enabling faster convergence of the PF. This proof of concept was subjected to a more comprehensive investigation in [9], where, for reasons of generality and relevance (cited above), the term *repair* was replaced by *progress*. The *IP operator* consists of the following modules.

1) *Training-Dataset Generation:* Here, the solutions from earlier generations are *mapped* to the current nondominated set.
2) *ANN-Training:* Here, an ANN is trained on the generated dataset to learn the *mapping model*.
3) *Offspring's Progression:* Here, the learnt ANN model is used to advance a fraction of the offspring in the current generation. Doing so enables these offspring to advance along the directions adopted by earlier solutions while evolving to the current set of nondominated solutions.

The following aspects relating to the IP operator are notable.

1) An EMO algorithm integrated with IP (EMO-IP) is different from (SAO) [9], as follows.
   a) While ML is used to learn the X-F relationships in SAO (for brevity, we avoid constraints here), it helps learn X-X relationships in the case of EMO-IP.
   b) In SAO, ML does not directly alter X, unlike in EMO-IP.
   c) In SAO, the offspring's evolution is done using genetic operators, while in EMO-IP, their output is subjected to ML-based alteration.
   d) In SAO, the solution evaluation is guided by approximate and actual F values, while EMO-IP relies only on the latter.
2) EMO-IP is different from local-search-assisted optimization (LSO) [9], as follows.
   a) In LSO, it is nontrivial to define a single objective in the presence of conflicting objectives, whereas, that is, not needed in EMO-IP.
   b) LSO requires additional solution evaluations in each generation, unlike EMO-IP.
3) The existing EOG methods and EMO-IP differ in their focus and computational implications as follows.
   a) EOG methods focus on producing better offspring, either by selecting a more promising set of parents, and/or by applying more appropriate mating strategies on a given set of parents. In contrast, EMO-IP relies on utilizing the history of gradual improvements in X-space (over an EMO algorithm's generations) toward *learning* the pertinent directional changes in X-space, and applying these to potentially improve the quality of offspring.
   b) The offspring, once created using the genetic operators, are not altered in EOG methods, unlike in EMO-IP.
   c) Evaluation of some extra solutions (using actual functions or surrogate models) is unavoidable in

EOG methods. In contrast, EMO-IP does not require any extra solution evaluation. This feature highlights a distinct advantage of EMO-IP over EOG methods.

4) In EMO-IP, the *offspring's progression* module can be considered as a part of the offspring creation process itself. The offspring are evaluated only after the *progression* module is executed, which requires no additional solution evaluations.

III. Enhanced Innovized Progress (IP2) Operator

It has been highlighted above that much like the IP operator, the IP2 operator also attempts to capture the directional improvements in the decision space to help the offspring advance effectively, through three modules, including: 1) *Training-Dataset Generation*; 2) *ML Training*; and 3) *Offspring's Progression*. The design and implementation of the constitutive modules of the IP2 operator are detailed below, followed by their integration with EMO/EMaO algorithms. Following these discussions, the manner in which these modules differ from those in the IP operator are highlighted in Section IV. In addition, the time and space complexities of each module are analyzed in Section S2 of the Supplementary Document (S.D.).

A. Training-Dataset Generation Module

At any generation t of the EMO/EMaO algorithm, let the final population in the preceding generation $(t-1)$, sized N, be referred to as the parent population P_t, and the offspring population (created from P_t), sized N, be denoted by Q_t. Let t_{past} denote a user-defined number of past generations that influences composition of the *input archive* A_t, whose members can be mapped onto the members of the *target-archive* T_t (of size N). The remainder of this section discusses the process of constituting and updating A_t and T_t, and archive *mapping*—where the solutions in A_t are mapped to ones in T_t, to yield the *training dataset*.

1) Input-Archive Composition and Update: At any generation t, the *input archive* A_t is intended to serve as a pool of reasonably diverse distinct solutions from previous generations, which can be mapped onto representative solutions in the current generation (*target-archive*), so that: 1) an ML method could *learn* the directional improvements in the decision space and 2) such a *learning* could be utilized to help some of the current offspring *progress* more effectively. In this spirit, $A_t = \{P_{t-t_{\text{past}}}\} \cup \{Q_{t-t_{\text{past}}}, Q_{t-t_{\text{past}}+1}, \ldots, Q_{t-1}\}$. Notably:

1) first both the parents and offspring in the $(t - t_{\text{past}})$th generation are included to account for maximum diversity without incorporating any duplicate solutions (since the parents and offspring in any generation are distinct);
2) in addition, only the offspring from the $(t - t_{\text{past}} + 1)$th until the $(t - 1)$th generation are included, while the corresponding parents are excluded. Such a choice is made considering that in any particular generation, not all parents may replaced by the offspring; hence, some of the parents in subsequent generations could be the same.

As the value of t increases, the A_t formulation above naturally offers the updated *input archive*.

2) Target-Archive Composition and Update: As indicated above, the *target-archive* T_t is intended to serve as a pool of diverse solutions onto which the solutions from previous generations (A_t) could be mapped to provide a basis for ML training and its subsequent use. In this spirit, T_t is defined as a set of N target solutions obtained along the N RVs, *till the* tth *generation*. This poses three pertinent questions.

1) How to initialize the *target-archive*, of size N, such that one target solution gets associated with each RV.
2) How to determine the potential targets from the parents in a subsequent generation t, namely, P_t.
3) How to update the existing *target-archive*, namely, T_{t-1}, by incorporating the potential targets from P_t. This, in effect, amounts to determining the best N target solutions *till the* tth generation.

The prerequisite for initialization of the *target-archive* T_t (at $t = 1$) and its subsequent update (at $t \geq 2$), is the normalization of the parent population P_t in the F-space. This normalization can be achieved using the ideal (Z^I) and nadir (Z^N) points.[1] Then, T_t can be initialized by associating one member of P_t *with each RV*, as its target. The criterion for such an association, wherever possible, is made to coincide with the metric adopted by the EMO/EMaO algorithm that the IP2 operator is to be integrated with. For instance:

1) given an RV, the solution offering minimum perpendicular distance (PD) [4] and penalty-based boundary intersection (PBI) [6] is to be chosen as the target, for NSGA-III and MOEA/D, respectively, (since these metrics are the ones employed by the respective algorithms);
2) in the case of an algorithm like NSGA-II, which is not RV based, the solution offering minimum achievement scalarizing function (ASF) [39] with a given RV is to be chosen as the target (benefits of using ASF in such a scenario could be found in [9]).

In any subsequent generation t, the following methodology is used to determine the potential targets for the RVs. *For each solution in* P_t, the relevant metric (ASF, PDM, PBI, or any other, depending on the underlying EMO/EMaO algorithm) is computed with respect to each RV. Notably, same RVs are used by both: 1) the IP2 operator and 2) underlying EMO/EMaO algorithm. Then, each solution in P_t is associated as the potential target for the RV offering minimum/best metric value. Clearly, some RVs may get associated with multiple potential targets, while some RVs may remain unassociated.

At any generation t ($t \geq 2$), the existing target archive T_{t-1} and the potential targets from P_t, are available. In this situation, the task of determining the best N target solutions *till the* tth *generation*, reduces to updating of T_{t-1} by accounting for the potential targets from P_t. In such a case, each RV may have two contenders for the target: 1) one, in the form of the existing target member from T_{t-1}, and 2) another, in

[1]Some base EMO/EMaO algorithms (including NSGA-III and MOEA/D) rely on the normalization of the F-space, whereas some (including NSGA-II) do not. For the former case, the IP2 operator uses the same Z^{I} and Z^{N} as estimated by the base algorithm. For the latter case, the IP2 operator employs the *simple normalization method* [38].

Algorithm 1 T_t = Update_Target_Archive (P_t, T_{t-1}, $\mathcal{R}$)

Require: Parent population P_t, target archive T_{t-1}, set of RVs $\mathcal{R}$
$\{F^T,\ F^P\} \leftarrow$ Objective function values in $\{T_{t-1},\ P_t\}$
Compute Z^{I} & Z^{N} from F^P `% ideal and nadir points`
$\{\bar{F}^T,\ \bar{F}^P\} \leftarrow$ Normalized $\{F^T,\ F^P\}$ using Z^{I} and Z^{N}
$V_{[N\times N]} \leftarrow \emptyset$ `% stores evaluated metric values`
$T_t \leftarrow T_{t-1}$
for i = 1 to N **do** `% for each solution in` $\bar{F}_P$
  for j = 1 to N **do** `% for each RV of` $\mathcal{R}$
    $V_{i,j} \leftarrow$ Metric value of $\bar{F}^P_{(i)}$ w.r.t. $\mathcal{R}_{(j)}$
  $V^P \leftarrow \min_{j=1}^{N} V_{i,j}$ `% best value for` i^{th} `solution`
  $I \leftarrow \arg\min_{j=1}^{N} V_{i,j}$ `% Index of RV for` i^{th} `solution`
  $V^T \leftarrow$ Metric value of $\bar{F}^T_{(I)}$ w.r.t. $\mathcal{R}_{(I)}$
  if $V^P < V^T$ **then**
    $T_{t,(I)} \leftarrow P_{t,(i)}$
return Updated target archive T_t

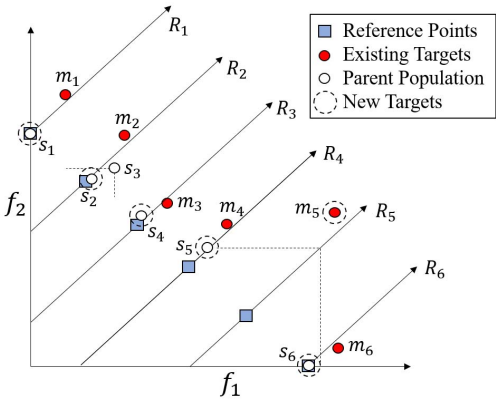

Fig. 1. Schematic for identifying target solutions along each RV, using ASF (with equal weights) in a two-objective space. Notice how a dominated solution (m_5) is considered as a target point for a poorly represented RV (R_5) for encouraging possible creation of points around that RV.

the form of the potential target from P_t. Again, the contender which fares better in the metric adopted by the underlying EMO/EMaO algorithm is declared as the updated target for that RV. The process of updating T_t is given in Algorithm 1.

Fig. 1 captures a realistic scenario for any generation t ($t \geq 2$), where, for each RV in $\mathcal{R} = \{R_1, R_2, \ldots, R_6\}$, there exists: 1) a corresponding target from $T_{t-1} = \{m_1, m_2, \ldots, m_6\}$ and 2) none, unique, or multiple potential targets from members in $P_t = \{s_1, s_2, \ldots, s_6\}$. Assuming that ASF is the underlying metric to be used, the update of the target-archive entails the following.

1) For each of R_1, R_3, R_4, and R_6, there is one associated target from T_{t-1}, and one potential target from P_t. As evident from the figure, the latter emerge as the updated targets, as they offer lower ASF values compared to the original targets from T_{t-1}.
2) For R_2, the associated target from T_{t-1} is m_2; however, there are two potential targets from P_t, namely, s_2 and s_3. Here, s_2 emerges as the updated target as it offers a lower ASF value compared to s_3 and m_2.
3) For R_5, the associated target from T_{t-1} is m_5, however, there are no potential targets from P_t. Hence, m_5 is retained as the updated target.

Algorithm 2 D_t = Archive_Mapping (A_t, T_t, $\mathcal{R}$)

Require: input archive A_t, target archive T_t, set of RVs $\mathcal{R}$.
$N_A \leftarrow$ Size of Archive A_t
$F^A \leftarrow$ Objective function values in A_t
Compute Z^{I} & Z^{N} from F^A `% ideal and nadir points`
$\bar{F}^A \leftarrow$ Normalized F^A using Z^{I} and Z^{N}
$D_t \leftarrow \emptyset$
$V_{[1\times N]} \leftarrow \emptyset$ `% stores evaluated metric values`
for i = 1 to N_A **do** `% for each solution in` A_t
  for j = 1 to N **do** `% for each RV in` $\mathcal{R}$
    $V_j \leftarrow$ Metric value of $\bar{F}^A_{(i)}$ w.r.t. $\mathcal{R}_{(j)}$
  $I \leftarrow \arg\min_{j=1}^{N} V_j$
  input $\leftarrow$ X-vector of $A_{t,(i)}$; *target* $\leftarrow$ X-vector of $T_{t,(I)}$
  $D_t \leftarrow D_t \bigcup$ [*input*, *target*]
return Training Dataset D_t

4) The *target archive* T_t, after factoring in both P_t and T_{t-1}, becomes $\{s_1, s_2, s_4, s_5, m_5, s_6\}$.

3) Archive Mapping: This is the last step of the *training-dataset generation* module, where the solutions in A_t are mapped onto those in T_t, to yield the *training dataset* D_t. Each solution in A_t is associated to some RV (as discussed above), and with each RV a solution from the T_t is associated (as discussed above). Hence, the involved RV provides a basis for association of each solution in A_t with a particular solution in T_t. For instance, if a solution $a_i \in A_t$ is associated with a particular RV, say R_j, then there is also a solution, say $m_k \in T_t$, that is, associated with R_j. Hence, the solution a_i gets mapped onto m_k. Since the goal of the IP2 operator is to learn the directional improvements in the decision (X)-space, only the X vectors of these solutions are stored in D_t. In the context of the above example, the X vectors of a_i and m_k together form one input-target sample for D_t. This process is shown in Algorithm 2.

Notably in Fig. 1, the solution m_5 symbolizes a larger issue of handling the *convergence-diversity tradeoff*. Since m_5 is dominated by other target members (s_5 and s_6), it characterizes poorer convergence. However, being the best-representative for R_5 till the tth generation, m_5 marks a suitable choice for maintaining diversity. In this background, the following options are open.

1) Both R_5 and m_5 be dropped, implying that no solutions in A_t associated with R_5 contribute to D_t.
2) R_5 be kept while m_5 is dropped, implying that solutions in A_t associated with R_5 can be mapped to a target associated with a nearby RV, say s_5 or s_6 (more converged than m_5).
3) Both R_5 and m_5 be kept, implying that the solutions in A_t associated with R_5 are mapped to m_5.

The first scenario would fail to capture any information about the solutions associated to R_5 in the *training dataset*, making it an unsuitable choice. The second scenario would amount to pursuing better convergence at the cost of loss in diversity. The third scenario would amount to attaching importance to diversity preservation during offspring's progression even at the cost of poorer convergence. In this article, a judicious choice in favor of the third scenario has been made, guided by the following rationale: 1) diversity preservation is crucial, especially in the early EMO/EMaO generations and 2) the marginal

Algorithm 3 (Predict(), $[\mathbf{x}^{\min}, \mathbf{x}^{\max}]$) = Training ($D_t$, $[\mathbf{x}^l, \mathbf{x}^u]$)

Require: training dataset D_t, lower & upper bounds of variables specified in the problem, $\mathbf{x}^l$ and $\mathbf{x}^u$.
1: $\{\mathbf{x}^{l,t}, \mathbf{x}^{u,t}\} \leftarrow$ Minimum and Maximum of each variable in D_t
2: $\mathbf{x}^{\min}, \mathbf{x}^{\max} \leftarrow \emptyset, \emptyset$ % bounds for dynamic normalization
3: **for** $k = 1$ to n_{var} **do** % Number of variables
4: $x_k^{\min} = 0.5(x_k^{l,t} + x_k^l)$
5: $x_k^{\max} = 0.5(x_k^{u,t} + x_k^u)$
6: Normalize D_t using $\mathbf{x}^{\min}$ and $\mathbf{x}^{\max}$ as bounds
7: Train the ML model using D_t to create Predict()
8: **return** Predict(), Bounds $[\mathbf{x}^{\min}, \mathbf{x}^{\max}]$

Algorithm 4 Q_t = Progress (Q_t,η,$[\mathbf{x}^{\min}, \mathbf{x}^{\max}]$,$[\mathbf{x}^l, \mathbf{x}^u]$

Require: offspring Q_t, jutting parameter η, bounds from Algorithm 3 $[\mathbf{x}^{\min}, \mathbf{x}^{\max}]$, variable bounds in problem definition $[\mathbf{x}^l, \mathbf{x}^u]$.
1: $I \leftarrow$ Randomly selected 50% of offspring Q_t
2: **for** $X \in I$ **do** % for each selected offspring
3: $\bar{X} \leftarrow$ Normalize X using $\mathbf{x}^{\min}$ and $\mathbf{x}^{\max}$
4: $\bar{X}^P \leftarrow$ Predict($\bar{X}$)
5: $X^P \leftarrow$ Denormalize $\bar{X}^P$ using $\mathbf{x}^{\min}$ and $\mathbf{x}^{\max}$
6: **for** each variable $k \in [1, n_{\text{var}}]$ **do**
7: Restore X_k^P to X_k if X_k lies in the vicinity of its bounds
8: $X^{PJ} \leftarrow$ Jutted offspring from X, X^P and η % equation 1
9: Boundary Repair on X^{PJ}
10: Replace the original offspring in Q_t by X^{PJ}
11: **return** Progressed offspring Q_t

compromise in convergence can always be overcome during subsequent generations as long as all regions of the search space are equitably explored.

B. ML Training Module

In this module, the *training dataset* D_t is used to train the ML model. As a preparatory step, D_t is normalized using a *dynamic normalization* method [9] that employs the bounds created using: 1) the lower and upper bounds for each variable as defined in the problem, i.e., $[\mathbf{x}^l, \mathbf{x}^u]$ and 2) the minimum and maximum of each variable in D_t, i.e., $[\mathbf{x}^{l,t}, \mathbf{x}^{u,t}]$. The final bounds employed for normalization, i.e., $[\mathbf{x}^{\min}, \mathbf{x}^{\max}]$ are estimated in lines 3–6 of Algorithm 3. Once the normalization is done, the ML model is trained as presented in Algorithm 3. Notably, each time the IP2 operator is invoked, a new ML model is trained and the last trained ML model is discarded.

Here, a different ML method than the ANN method used earlier—namely, RF—is employed, to provide an alternative way to learn the input-target patterns in the *training dataset*. The input and target in each sample in D_t, are vectors of size n_{var} (number of variables) each. The RF has three critical parameters to be defined—i.e., the number of trees (n_{trees}), number of variables/features considered while splitting a node (n_{features}), and the splitting criterion. Considering that the size of the training dataset is N_{TD}, the parameters are fixed as: $n_{\text{trees}} = N_{TD}$ and $n_{\text{features}} = n_{\text{var}}$. The splitting criterion used is *Mean Squared Error* (MSE), and the rest of the RF settings are kept as default.[2]

C. Offspring's Progression Module

In this module, the ML model trained on D_t is applied to help some of the offspring Q_t in the current generation to advance in useful directions, as discussed as follows.

1) *ML Model-Application*: A randomly selected 50% offspring are advanced using the trained ML model. It is notable that the offspring are normalized and denormalized (before and after advancement, respectively) using the same bounds created for *dynamic normalization* in *ML training*.
2) *Near-Bound Restoration:* While the entire X-vector of an offspring undergoes advancement, care is taken to observe those particular variables which before advancement happened to be within a 1% vicinity of their respective bounds. For such specific variables, their original values are restored. The rationale behind this is explained in [9].
3) *Jutting the Advanced/Progressed Offspring:* Understandably, the ML model *learns* the directional improvements that helped the solutions from previous generations transition to the best solution found along each RV till the current generation. The current offspring's advancement using such an ML model is based on the assumption that the directions found pertinent in the past shall again help the offspring transition to better solutions. In this situation, this article treats the *learnt* directions for different RVs as promising search directions, and introduces the notion of step length through the parameter η, leading to jutted offspring solutions, given as follows:

$$X^{PJ} = X + \eta \times \left(X^P - X\right) \tag{1}$$

where X and X^P mark the original and progressed offspring, respectively, and X^{PJ} represents the jutted offspring. Notably, $\eta = 1$ leads to the naturally progressed offspring X^P, while $\eta > 1$ leads to jutted offspring. At a more fundamental level, jutting counters the limitation that many ML-based regression methods, including RF, are not suitable for extrapolation, despite their excellence in predicting the data that can be interpolated from the input training dataset. In effect, the challenge that IP2 operator may not be helpful in creating offspring in regions that can only be achieved by extrapolating on the current population, is largely alleviated by jutting.
4) *Boundary Repair:* If any variable $\mathbf{x}_k \in X^{PJ}$, post advancement, goes outside its permissible bounds $[\mathbf{x}_k^l, \mathbf{x}_k^u]$, it is mapped to an inner value based on an inverse parabolic spread distribution [40].

The overall process of the offspring's progression module is presented in Algorithm 4. A sample illustration is shown in Section S4 of S.D.

D. Integration With EMO/EMaO Algorithms

This section highlights the integration of the IP2 operator within the architecture of an EMO/EMaO algorithm. The

[2]The RF Regressor used in this study has been taken from the Scikit-learn implementation (for python language).

Algorithm 5 Generation t of NSGA-II/III With IP2 Operator

Require: set of RVs $\mathcal{R}$, parent population P_t, input archive A_t, existing target archive T_{t-1}, variable bounds defined in problem $[\mathbf{x}^l, \mathbf{x}^u]$, parameters: t_{past}, t_{freq} and η.

1: $T_t \leftarrow$ Update_Target_Archive($P_t, T_{t-1}, \mathcal{R}$)
2: $count \leftarrow \text{rem}(t, t_{\text{freq}})$
3: **if** $count = 0$ **then**
4: $\quad flag \leftarrow$ True
5: **else**
6: $\quad flag \leftarrow$ False
7: **if** $flag$ =True **then**
8: $\quad D_t \leftarrow$ Archive_Mapping($A_t, T_t, \mathcal{R}$)
9: $\quad$ (Predict, $[\mathbf{x}^{\min}, \mathbf{x}^{\max}]$) $\leftarrow$ Training($D_t, [\mathbf{x}^l, \mathbf{x}^u]$)
10: $Q_t \leftarrow$ Crossover + Mutation(P_t)
11: **if** $flag$ = True **then**
12: $\quad Q_t \leftarrow$ Progress(Q_t,η,$[\mathbf{x}^{\min}, \mathbf{x}^{\max}]$,$[\mathbf{x}^l, \mathbf{x}^u]$,Predict)
13: Evaluate Q_t
14: $A_{t+1} \leftarrow (A_t \cup Q_t \cup P_{t+1-t_{\text{past}}}) \setminus [P_{t-t_{\text{past}}} \cup Q_{t-t_{\text{past}}}]$
15: $P_{t+1} \leftarrow$ Survival selection of NSGA-II/III with $P_t \cup Q_t$
16: **return** P_{t+1}, A_{t+1}, T_t

IP2's integration involves the introduction of another parameter, namely, t_{freq}, that specifies the number of generations between two successive offspring's progressions. The integration of the IP2 operator with NSGA-II and NSGA-III is detailed below. Further details, including integration of the IP2 operator with MOEA/D and MaOEA-IGD, are given in Section S3 of S.D.

NSGA-II and NSGA-III: The integration of the IP2 operator is presented in Algorithm 5. The pseudocode represents an intermediate generation of these algorithms, and the integration of the IP2 modules therein. First, the *target archive* T_t is updated using Algorithm 1. Then, guided by t_{freq}, it is assessed whether or not the IP2 operator is to be activated (lines 2–6). If yes, the flag is marked *True*, else *False*. Then the *training dataset* D_t is generated using Algorithm 2, and the ML training is done using Algorithm 3 (lines 7–9). Subsequently, offspring are produced using the genetic operators (crossover and mutation, line 11), followed by: 1) offspring's progression using Algorithm 4; 2) offspring's evaluation (line 14); 3) update of the *archive* A_t,; and 4) survival selection (line 15).

IV. Differences Between IP and IP2 Operators

This section aims to highlight the key differences between the IP and IP2 operators, with respect to the design and implementation of each of their constitutive modules. Notably, despite these differences, their basic architecture is the same. Hence, IP2 also differs from the SAO, LSO, or EOG methods, as cited in Section II-B with respect to the IP operator.

A. Training Dataset Generation

Here, the differences are in the manner in which the *input archive* and *target archive* are individually composed, in the case of IP and IP2 operators, owing to which the resulting archive mapping also differs between the operators.

1) *Input Archive:* The IP operator stores the parent solutions from t_{past} previous generations in the *input archive*. Since not all parents may get replaced by offspring at every generation, the *input archive* contains duplicates—more so in later generations, where a smaller fraction of offspring are likely to outperform the parents and be selected. Duplicates in the *input archive*, and also in the *training dataset*, can undermine the scope of the subsequent ML learning and its application toward progression of the offspring. In contrast with the IP operator, the above challenge of duplicates is inherently overcome by the manner in which the input archive is constituted for the IP2 operator, promising better ML assistance to EMO/EMaO algorithms.
2) *Target Archive:* The IP operator selects the target solutions for each RV only from the nondominated solutions in the current parent population. In case no nondominated solution gets associated with an RV, a nondominated solution associated with a different RV is selected. The essence is that the IP operator singularly pursues better convergence properties in the target archive. In contrast, the IP2 operator provides for a better convergence-diversity tradeoff by allowing for such solutions to become the targets for some RVs, which may be dominated by other members in the target archive. (Section III-A3).

B. ML Training

The proposed IP2 operator uses RF as the ML method, while ANN was used with the original IP operator. While training an RF is known to be computationally more efficient than training an ANN [14], the quantitative impact of adoption of RF over ANN, in terms of the performance of EMO algorithms, has also been investigated in Section VI-A.

C. Offspring's Progression

Here, the differences in context of the offspring *progression* module are highlighted.

1) *Boundary Repair:* In the IP operator, if any variable goes out of its bounds, it is fixed at the respective boundary value. However, in IP2, the value is determined using an inverse parabolic spread distribution [40]. This is beneficial since fixing a variable to its boundary in multiple solutions may cause inefficient mating.
2) *Jutting the Progressed Offspring:* This is a new proposition in the IP2 operator. It controls the extent to which the ML model is exploited for the offspring's progression—the direction is determined by the RF, but how far the offspring are moved in that direction is controlled by jutting. The effect of jutting is visualized through a small parametric study in Section V-C.

V. Experimental Settings

This section presents the test problems and performance indicator used, the parameter settings for the IP and IP2 operators, underlying EMO/EMaO algorithms, and the details on the reference-point generation for the latter.

A. Test Suite

For multiobjective test instances, two-objective ZDT [16] and three-objective DTLZ [17], WFG [18], and MaF [19] problems are used with the following specifications.

1) *ZDT:* Their respective $g(X)$-functions have been modified to have PO solutions at $x_k = 0.5$, for $k = 2, \ldots, 30$ [9].
2) *DTLZ:* The total number of variables are kept as 15.
3) *WFG:* The position-related variables k are kept as $k = 2 \times (M - 1)$, and the distance-related variables in are kept at 20. The other parameters controlling the difficulty levels in these problems are set to the default values given in [18].
4) *MaF:* The distance variables are kept as 20.

Once the comparison between the IP and IP2 operators is presented with respect to the above problems, more insights into the efficacy of the IP2 operator are shared with respect to two other two-objective test problems with linked variables, namely, *L1* and *L2* ($n_{\text{var}} = 10$) [9].

For many-objective test instances, the 5-, 8-, and 10-objective variants of the following problems are used.

1) *DTLZ Problems:* Here, the default value of position-related variables is used as $k = (M - 1)$, while the distance-related variables are set as 20.
2) *WFG and MaF Problems:* Here, the settings used are the same as those for multiobjective instances.

B. Performance Indicator and Statistical Analysis

For performance comparison, a hypervolume (HV) measure that accounts for both convergence and diversity has been used. For each value of M, the choice of reference point ($\vec{r}$) for HV calculation (and the rationale behind it) is given in Section S5 of S.D. In that, for estimating $\vec{r}$, an existing method [41] (suited only for RVs created using the Das–Dennis method) is extended for RVs created using a different method as well. Notably:

1) while for the DTLZ problems, the scales of different objectives are equal, they differ for WFG and some MaF problems. Hence, for the latter, the solutions are normalized in F-space using the theoretical PF extremes;
2) when comparing *only two* algorithms, at a time (as in the case of many-objective problems), the Wilcoxon ranksum test [42] is performed on the HV values reported over multiple/independently seeded runs. Here, the threshold p-value of 0.05 (95% confidence level) is used;
3) when comparing *more than two* algorithms, at a time (as in multiobjective problems), the Kruskal–Wallis test [43] with threshold p-value of 0.05 is used, to infer if their overall differences are statistically insignificant or not. If not, the Wilcoxon ranksum test [42] is used for their pairwise comparisons, where the algorithm reporting the best median HV is treated as a reference. Furthermore, the threshold p-value is adjusted using the standard Bonferroni correction [44], to retain the same overall confidence. More details on these tests, are presented in Section S6 of S.D;
4) for visualizing the magnitude of improvement in HV, Cohen's effect size (d_e) is also reported for all experiments (further detailed in Section S7 of S.D.).

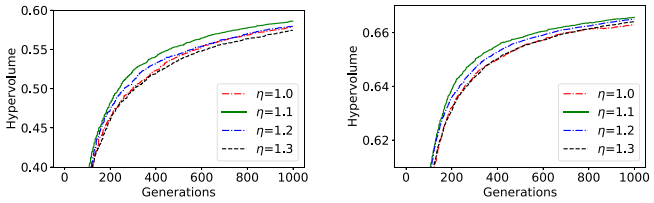

Fig. 2. Median HV for different η on problems *L1* and *L2*. (a) Problem *L1*. (b) Problem *L2*.

C. Parameter Settings

1) Parameters for the IP and IP2 Operators: Both the IP and IP2 operators use parameters, namely, t_{past} and t_{freq}, both of which are set as 5 (aligned with the settings in [9]). A detailed discussion on t_{past} is presented in Section S8 of S.D. The IP2 operator additionally involves a parameter η, which is set as 1.1. While a more robust criterion for choosing η may need to be identified, this choice is based on a preliminary experimental investigation with $\eta = 1.0, 1.1, 1.2$, and 1.3 on a pair of relatively difficult (with linked variables) problems, namely, the *L1* and *L2* problems. In that situation, the median HV over 16 runs with 1000 generations in each run are shown in Fig. 2. It is evident that in this case, $\eta = 1.1$ offers a consistent advantage over the other alternatives tested.

Besides the above parameters, additional parameter settings related to the underlying ML method, namely, ANN and RF are required. While the settings for the ANN architecture and training are taken from [9], the settings for RF are as discussed earlier in Section III-B.

2) Parameters for the EMO/EMaO Algorithms: For all the multiobjective instances, and the four EMO/EMaO algorithms involved, the presented results correspond to 16 independently seeded runs, with the following parameter settings.

1) The population size N is kept as 100 and 105 for $M = 2$ and 3, respectively, and reference-points are generated using the Das–Dennis method [45].
2) The SBX crossover ($p_c = 0.9$ and $\eta_c = 10$) and polynomial mutation ($p_m = 1/n_{\text{var}}$ and $\eta_m = 20$) are used.

Notably, MOEA/D involves additional parameters, the settings for which include: 1) scalarization metric = PBI; 2) neighborhood size $T = 10\%$ of N; 3) probability of neighborhood mating $\delta = 0.8$; 4) replacement size $nr = 2$.

For all many-objective instances, and the three EMaO algorithms involved, all the parameter settings stated above hold, with the exception that an alternative method for generating the reference points (discussed below) has been used, leading to a customized population size N for a given value of M.

3) Reference Point Generation in Many-Objective Problems: A recent study [46] revealed that with the Das–Dennis method, the proportion of boundary points escalates rapidly as M increases. This method takes as input the number of gaps p and number of objectives M. In this case:

1) *for $p < M$:* no interior points are generated;
2) *for $p = M$:* exactly one interior point is generated;

TABLE I
PARAMETERS FOR GENERATING REFERENCE POINTS USING THE LAYER-WISE s-ENERGY METHOD VERSUS THE DAS–DENNIS METHOD. FOR THE EXPERIMENTS, DAS–DENNIS IS USED FOR $M = 3$ AND LAYER-WISE s-ENERGY METHOD IS USED FOR $M = \{5, 8, 10\}$

M	Layerwise s-energy			Das-Dennis		
	p	N	% Boundary Points	p	N	% Boundary Points
3	{11, 9, 7, 5, 3}	105	31%	13	105	37%
5	{10, 8, 7, 6, 5}	335	28%	7	330	95%
8	{4, 3, 4, 3, 4}	404	22%	5	792	100%
10	{3, 3, 3, 3, 3}	500	20%	4	715	100%

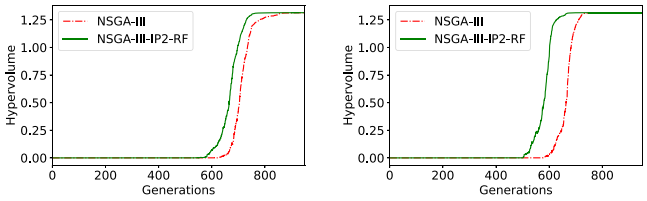

Fig. 3. DTLZ1 ($M = 5$): Plots for median HV reported by NSGA-III and NSGA-III-IP2-RF. (a) Das–Dennis method. (b) s-energy method.

3) *for $p > M$:* more interior points can be generated.

Understandably, for ML-assisted optimization to be effective, it is desired that the data set on which the ML model is trained contains a reasonable representation of both the boundary and interior points, implying the need for $p > M$. However, $p = M = 8$ leads to $N = 6435$, while $p = M = 10$ leads to $N = 92378$, which pose unreasonably large sizes. Hence, the following combinations of (M, p): $(3, 13)$, $(5, 7)$, $(8, 5)$, and $(10, 4)$ were explored, and the corresponding percentages of boundary points offered by the Das–Dennis method are presented in Table I. Clearly, the Das–Dennis method may not be a good choice for $M \geq 5$, implying the need for an alternative reference-point generation method.

A promising alternative is found in the layer-wise s-energy method [46], which first generates points using a layer-wise Das–Dennis method, then optimizes the shrinkage factor for each layer using the s-energy method. Table I highlights the potential benefits of using this method when applying the IP2 operator on many-objective problems, as it provides for reasonable population sizes along with a fine balance between boundary and interior points.

The DTLZ1 problem is used with $M = 5$, as an example to demonstrate how having a reasonable balance between boundary and interior points may be advantageous for the efficacy of ML-assisted optimization. The performance of the base NSGA-III is compared with NSGA-III-IP2-RF for two scenarios, where the underlying reference points for both, are generated by the Das–Dennis method (Fig. 3(a)) and the s-energy method (Fig. 3(b)). It can be seen that the difference in the performance of NSGA-III and NSGA-III-IP2-RF is more significant when the s-energy method is used. As evident from Fig. 3, the base EMO/EMaO algorithm alone may eventually achieve a good PF approximation. Hence, the search efficacy infused by the integration of IP2-RF should be judged, not just by the final value of the performance indicator, but also by how early in the generations, the improvements in the performance indicator are observed.

In wake of the above, the reference-points for problems with $M \geq 4$ are generated using the layer-wise s-energy method, for which the parameters are given in Table I. For $M < 4$, the Das–Dennis method is used. These reference points are used for both: 1) the IP2 operator and 2) the underlying EMaO algorithm—i.e., NSGA-III, MOEA/D, and MaOEA-IGD.

VI. EXPERIMENTAL RESULTS

This section aims to investigate the search efficacy infused into the EMO/EMaO algorithms through integration of the proposed *enhanced IP* operator, namely, IP2-RF. Considering that IP-ANN was proposed in an earlier study, the above aim is pursued through incremental investigations.

1) For multiobjective instances, the comparative impact of:
   a) the ML methods (ANN versus RF) is assessed, while retaining the same operator (IP);
   b) the operators (IP versus IP2) is assessed, while retaining the same ML method (RF).
2) For many-objective instances, the impact of integration of the IP2-RF operator with EMaO algorithms has been exclusively investigated.

A. Performance Comparison of the ML Methods: ANN Versus RF, With the IP Operator

In this section, for each of the following algorithms: NSGA-II, NSGA-III, and MOEA/D, their base version is compared with two variants, in which the base version is integrated with IP-ANN and then with IP-RF. The median HV values at generation $t = 100$ are shown in Table S9 (in S.D.) (here, columns 5, 9, and 13 with reference to IP2-RF may be overlooked), where one can see:

1) performances of NSGA-II-IP-ANN and NSGA-II-IP-RF are statistically similar in 2 (out of 5) test instances.
2) performances of NSGA-III-IP-ANN and NSGA-III-IP-RF are statistically similar in 16 (out of 19) test instances.
3) performances of MOEA/D-IP-ANN and MOEA/D-IP-RF are statistically similar in 24 (out of 24) test instances.
4) performances of MaOEA-IGD-IP-ANN and MaOEA-IGD-IP-RF are statistically similar in 21 (out of 24) test instances.
5) overall, the IP-RF variants are either better or statistically similar to the corresponding IP-ANN variants, in 68 instances out of 72.
6) noticeably, the performance of the IP operator is not very sensitive to the choice of the ML method (RF or ANN).

As highlighted in [14], training an RF is computationally more efficient than training an ANN. This is empirically validated (for a sample illustration), by the run time, in seconds, shown in brackets adjacent to the HV measures corresponding to the IP-ANN and IP-RF variants of NSGA-II and NSGA-III.

Considering that RF yields better or statistically similar results to those of ANN, in significantly less run time, it becomes a suitable choice for the IP2 operator in this article.

TABLE II
PERFORMANCE OF IP-ANN, IP-RF, AND IP2-RF OPERATORS, COMPARED INDEPENDENTLY FOR NSGA-II, NSGA-III, MOEA/D MAOEA-IGD. EACH COLUMN SHOWS THE MEDIAN HV (FROM 16 RUNS) AT GENERATION $t = 100$ FOR $M = 2$ (ZDT) AND $M = 3$ (DTLZ, WFG, AND MAF) PROBLEMS, WITH THE BEST VALUE AND ITS STATISTICAL EQUIVALENT, MARKED IN BOLD. THE COMPUTATIONAL TIME (IN SECONDS) WITH DIFFERENT ML METHODS (ANN AND RF) IS SHOWN IN BRACKETS "()." THE TOTAL IN EACH COLUMN INDICATES THE COUNT OF INSTANCES WHEN THE PERFORMANCE WAS BEST OR STATISTICALLY EQUIVALENT TO THE BEST

Problem	NSGA-II	NSGA-II-IP-ANN	NSGA-II-IP-RF	NSGA-II-IP2-RF	MOEA/D	MOEA/D-IP-ANN	MOEA/D-IP-RF	MOEA/D-IP2-RF	MaOEA-IGD	MaOEA-IGD-IP-ANN	MaOEA-IGD-IP-RF	MaOEA-IGD-IP2-RF
ZDT1	0.67543	0.67731 (283)	0.67885 (131)	**0.67900**	**0.04376**	**0.05314**	**0.03784**	**0.05184**	**0.32759**	0.24444	**0.27878**	**0.29598**
ZDT2	0.34047	0.34282 (247)	0.34435 (124)	**0.34522**	0.03481	0.03033	0.00690	**0.10379**	**0.13308**	**0.12809**	**0.12131**	**0.12940**
ZDT3	**0.53203**	**0.53364 (301)**	**0.53439 (132)**	**0.53452**	**0.13673**	**0.18619**	**0.17071**	**0.21108**	**0.61120**	**0.61054**	0.57916	**0.62713**
ZDT4	**0.67348**	**0.67601 (278)**	**0.67409 (129)**	**0.67696**	**0.00000**	**0.01894**	**0.02856**	**0.00588**	0.42885	0.37266	0.42497	**0.43534**
ZDT6	0.15920	0.16472 (256)	0.16663 (126)	**0.22850**	**0.01336**	**0.00958**	**0.01102**	**0.01830**	**0.16276**	**0.15562**	**0.13347**	**0.14635**
	NSGA-III	NSGA-III-IP-ANN	NSGA-III-IP-RF	NSGA-III-IP2-RF	–	–	–	–	–	–	–	–
DTLZ1*	1.17915	1.19079 (236)	1.18517 (97)	**1.21666**	**1.03030**	**1.04424**	**1.03955**	**1.04344**	**1.19107**	**1.16481**	**1.16012**	**1.18204**
DTLZ2	0.65207	**0.65913 (224)**	**0.65792 (95)**	**0.65747**	0.36164	0.34708	0.35063	**0.41710**	0.33690	0.37799	0.37146	**0.39629**
DTLZ3*	0.66977	0.38448 (219)	0.65693 (98)	**0.87723**	**0.59594**	**0.53397**	**0.62223**	**0.56537**	**1.03353**	**1.00473**	**0.96886**	**0.94684**
DTLZ4	0.65284	**0.65833 (228)**	**0.65831 (97)**	**0.65866**	**0.34177**	**0.30583**	**0.31773**	**0.29841**	0.25757	0.24591	**0.29619**	**0.29325**
WFG1	**0.14763**	**0.12956 (443)**	**0.12753 (114)**	**0.13292**	**0.13437**	**0.15434**	**0.14385**	**0.17619**	0.00249	0.00635	0.00671	**0.01115**
WFG2	**1.06465**	**1.08261 (432)**	**1.06723 (121)**	**1.07296**	**0.74989**	**0.74523**	**0.72235**	**0.76342**	**0.80138**	0.76571	**0.78154**	**0.79363**
WFG4	0.55477	0.57084 (453)	0.57098 (114)	**0.57591**	0.37159	**0.38210**	**0.37985**	**0.40120**	0.49249	0.48736	0.50904	**0.56400**
WFG5	**0.50904**	**0.50580 (440)**	**0.50297 (115)**	**0.50536**	**0.29559**	**0.30140**	**0.29488**	**0.29231**	0.36941	**0.40133**	0.32237	**0.39148**
WFG6	0.50206	0.53764 (458)	**0.54403 (122)**	**0.55756**	0.26679	0.27658	0.27352	**0.30046**	**0.48231**	**0.47698**	**0.45346**	**0.48760**
WFG7	0.57459	0.56950 (439)	0.55137 (119)	**0.58404**	0.28008	0.27480	0.27131	**0.31635**	0.37537	0.40673	0.42340	**0.47627**
WFG8	**0.45872**	**0.45000 (451)**	**0.44266 (121)**	**0.45233**	**0.25884**	**0.25415**	**0.25941**	**0.27089**	0.23380	0.35692	0.31303	**0.40388**
WFG9	0.52862	0.54274 (431)	0.54870 (124)	**0.56847**	0.25648	0.24939	0.25450	**0.30935**	**0.21117**	**0.22255**	**0.20126**	**0.23813**
MaF1	**0.22065**	**0.22530 (225)**	**0.22548 (98)**	**0.22812**	0.11909	0.11230	0.10551	**0.13094**	**0.16809**	**0.16824**	**0.16869**	**0.16745**
MaF2	0.38853	**0.39351 (231)**	**0.39354 (95)**	**0.39314**	0.32391	0.32231	0.32008	**0.33141**	**0.30256**	**0.30575**	**0.29373**	**0.30089**
MaF3*	0.00000	0.00000 (225)	0.00164 (103)	**0.01665**	**0.04701**	**0.00000**	**0.02317**	**0.00000**	**0.12372**	0.00000	0.03569	0.01399
MaF4*	0.02460	0.02891 (218)	**0.06793 (98)**	**0.06694**	**0.12317**	0.05668	0.07839	0.09345	**0.28741**	**0.22559**	**0.26249**	**0.17393**
MaF5	**1.13214**	**1.12242 (242)**	**1.12328 (96)**	**1.07268**	**0.65706**	**0.64143**	**0.63111**	**0.70548**	0.95133	**0.97297**	0.95383	**0.97697**
MaF7	0.33331	0.33571 (237)	0.34363 (98)	**0.35610**	**0.04863**	**0.04975**	**0.04074**	**0.05079**	0.20412	**0.25529**	**0.24345**	0.22847
MaF13	**0.36061**	**0.36059 (235)**	**0.36061 (95)**	**0.36656**	0.09681	**0.11324**	0.09878	**0.12538**	0.23772	**0.26631**	0.19353	**0.27197**
Total ⟶	07/19	10/19	12/19	**19/19**	15/24	16/24	15/24	**23/24**	13/24	14/24	13/24	**22/24**

* The HV values are calculated using a custom reference-point since the solutions haven't converged to the hypercube defined by the PF.

B. Performance Comparison of the Progress Operators: IP Versus IP2, With RF as the Underlying ML Method

This section aims to compare the IP and IP2 operators, toward which a comparison is drawn between the IP-RF and IP2-RF variants for each of, NSGA-II, NSGA-III, MOEA/D, and MaOEA-IGD. Hence, attention may be paid to columns 4, 5, 8, 9, 12, and 13 of Table II, observing that:

1) NSGA-II-IP2-RF performs statistically better than or similar to NSGA-II-IP-RF, in 5 out of 5 instances;
2) NSGA-III-IP2-RF performs statistically better than or similar to NSGA-III-IP-RF, in 19 out of 19 instances;
3) MOEA/D-IP2-RF performs statistically better than or similar to MOEA/D-IP-RF, in 23 out of 24 instances;
4) MaOEA-IGD-IP2-RF performs statistically better than or similar to MaOEA-IGD-IP-RF, in 22 out of 24 instances.

Overall, the IP2-RF variants are either statistically better than or similar to the corresponding IP-RF variants, in 69 instances out of 72. This could directly be attributed to the enhancements in the constitutive modules of the IP operator that lead to IP2. For further insights, the effect sizes (d_e) are reported in Section S11-A of S.D. Comparison of the base version of each EMO algorithm with its respective variant using IP2-RF reveals a distinct advantage (in terms of the HV measure) of the latter, and builds a case for ML-assisted optimization, particularly when the advantages can be achieved without incurring any additional solution evaluations.

C. Impact of the IP2-RF Operator on Many-Objective Problems

This section aims to investigate whether the case for ML-assisted optimization established above in the context of multiobjective problems also extends to more difficult many-objective problems. Toward this end, the proposed and so far best-performing IP2-RF operator is integrated with EMaO algorithms NSGA-III, MOEA/D, and MaOEA-IGD, and the performance is compared with the respective base versions. This comparison is done at $t = 500$, with reference to the DTLZ, WFG, and MaF problems involving $M = 5$, 8, and 10. The median HV values, along with their effect sizes (d_e), shown in Table S9, in the supplementary material, reveal the following.

1) In terms of HV, all EMaO algorithms, including NSGA-III-IP2-RF, MOEA/D-IP2-RF, and MaOEA-IGD-IP2-RF offer a distinct advantage over their base versions.
2) In most of the test instances considered, the improvement in HV is moderate ($d_e > 0.5$) or large ($d_e > 0.8$) [47].
3) Out of the 171 instances, the EMaOs integrated with the IP2-RF operator perform statistically worse than their base versions ($d_e < 0$) in only $\sim 6\%$ (11 out of 171) instances.
4) The (marginal) differences in the trends obtained with NSGA-III-IP2-RF, MOEA/D-IP2-RF, and MaOEA-IGD-IP2-RF, respectively, could be attributed to the heuristics employed by the underlying EMaO

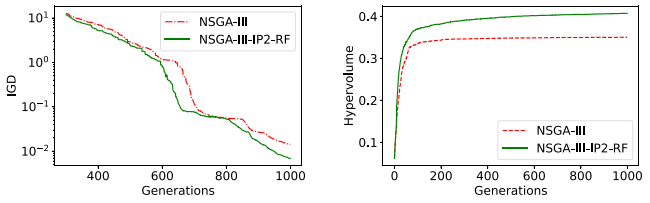

Fig. 4. Median IGD plot for DTLZ3 and median HV plot for WFG9 (multimodal problems). (a) DTLZ3 ($M = 3$). (b) WFG9 ($M = 3$).

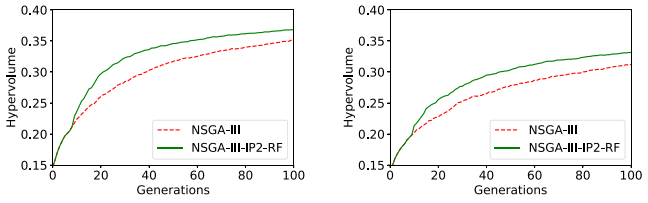

Fig. 5. Median HV plots for WFG4. (a) WFG4 ($M = 3$). (b) Modified WFG4 ($M = 3$).

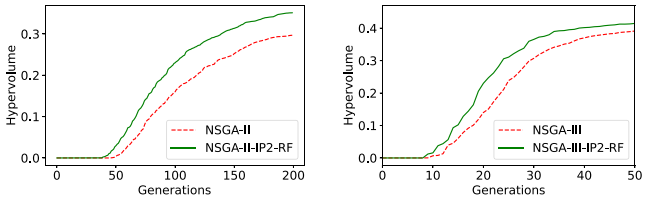

Fig. 6. Median HV plots for biased problems. (a) ZDT6 (default: $M = 2$). (b) DTLZ4 ($M = 3$).

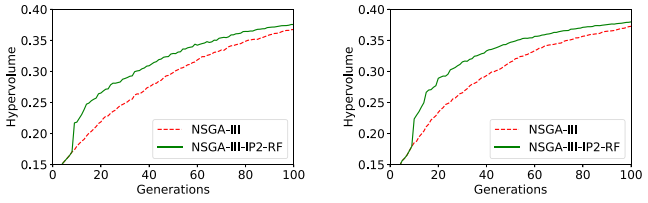

Fig. 7. Median HV plots for WFG7. (a) WFG7 ($M = 3$). (b) Modified WFG7 ($M = 3$).

algorithms, given which the characteristics of the training data-sets may change.

Based on the totality of experiments on both multi- and many-objective problems, it is fair to infer that integration of the base EMO/EMaO algorithms with the IP2-RF operator shows a lot of promise for performance enhancement, without incurring additional solution evaluations or unreasonably high computational time for ML training.

VII. IP2-RF Operator: From Performance Trends to Insights

In the previous section, the focus was on highlighting the comparative performance trends between the base EMO/EMaO algorithms and their variants incorporating IP2-RF. Adding to it, a scalability study of the IP2 operator, with respect to: 1) the number of variables (n_{var}) and 2) the number of objectives (M), is presented in Section S9 of S.D. Here in this section, the aim is to focus a little more deeply and share insights on the problem characteristics that generally pose difficulties for the base algorithms, and hence, offer the potential for performance enhancement by IP2-RF. This section has a dedicated focus on highlighting the performance of the IP2-RF operator on problems characterized by: *multimodality*, *bias*, *intervariable linkages*, and *discrete variables*. This is followed by an investigation of the IP2 operator on four real-world problems. In addition, results on some special test problems (with small numbers of variables) are presented in Section S11-B of S.D.

A. Multimodality of Pareto-Optimal Set

Some test problems exhibiting multimodality are DTLZ3, WFG4 and WFG9 [18]. The generation-wise median IGD plot of DTLZ3 and median HV plot of WFG9 are shown in Fig. 4(a) and (b), respectively. In Fig. 4(a), it is evident that the performance of NSGA-III-IP2-RF is consistently better than or on a par with that of NSGA-III. Similarly, Fig. 4(b) shows that the use of the IP2-RF operator helps NSGA-III achieve better convergence and diversity.

WFG4 is also multimodal, with difficulty that depends on three parameters: $\{A, B, C\} = \{30, 10, 0.35\}$. Higher values of A and smaller values of B create a more difficult problem [18]. Hence, a modified-WFG4 is solved here, with $\{A, B, C\} = \{70, 5, 0.35\}$. The HV plots for both WFG4 and this modified-WFG4 are shown in Fig. 5. Both HV values in Fig. 5(b) are comparatively lower than their corresponding values in Fig. 5(a), but NSGA-III-IP2-RF still yields a performance improvement on the difficult version of WFG4.

B. Bias in the Pareto-Optimal Set

Some known test instances with bias are ZDT6, DTLZ4, and WFG7 [18]. The generation-wise median HV plots for problems ZDT6 and DTLZ4 are shown in Fig. 6(a) and (b), respectively. The advantage of using the proposed IP2-RF operator is again evident from these plots. Theoretically, it makes sense to have the advantage of a learning-based operator in biased problems, since the learning will help reach the optimal distribution of solutions faster once some representative points are identified.

As in the multimodal problem WFG4, the parameters of the WFG7 problem can also be modified, since it has a parameter-dependent bias. The parameter C in WFG7, which is 50 [18], is here increased to 100, to yield the modified-WFG7 problem. The HV plots for both versions, shown in Fig. 7, reveal that the IP2-RF operator does infuse better search efficacy.

C. Linked Variable Values in the Pareto-Optimal Set

It is known that common genetic operators, such as SBX crossover and polynomial mutation operators, do not account for any intervariable relationships or linkages. An example crossover operator that can preserve variable linkages

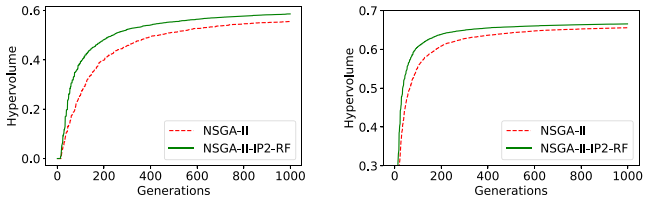

Fig. 8. Median HV plots for *L1* and *L2* problems. (a) *L1*. (b) *L2*.

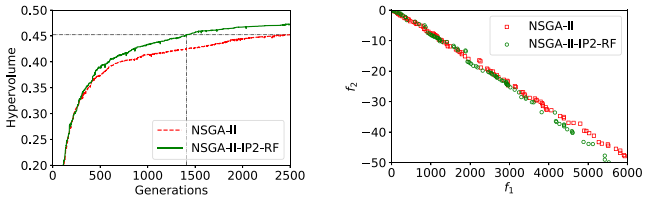

Fig. 9. Gearbox design problem, showing better performance by NSGA-II-IP2-RF. (a) Median HV plot. (b) Median front at $t = 2500$.

is the operator from differential evolution (DE). However, in problems with independent variables, DE is not as efficient in gaining convergence [9]. Given this background, it is intriguing to investigate whether IP2-RF could favorably assist an EMO algorithm employing the SBX crossover operator. Toward that end, two problems *L1* and *L2* [9] with linked variables are solved using NSGA-II and NSGA-II-IP2-RF. The generation-wise HV plots are presented in Fig. 8(a) and (b). These plots clearly depict the advantage of using the IP2-RF operator in such scenarios.

The variable linkages present in test instances are known, *a priori*, hence, one is able to choose judiciously a suitable crossover (DE or SBX) operator. However, such information may not be available in most real-world problems. In such a scenario, integrating the base EMO with the IP2-RF operator could be an effective approach.

D. Discrete-Variable (Real World) Problem

Notably, all the results presented above relate to test problems involving continuous variables only. However, there are many real-world problems that involve discrete variables as well. Hence, it is intriguing to investigate whether the benefits observed for continuous problems could also translate to discrete problems. For a sample illustration, the multi-speed gearbox design problem [48], involving 24 discrete variables,[3] is considered here. Notably, the objectives relate to minimization of overall gear volume (f_1), maximization of power delivered (f_2), and minimization of center-to-center distance between shafts (f_3). However, since f_1 and f_3 are correlated [49], a two-objective version of the problem (with f_1 and f_2 only) is solved here.

Mating for discrete variables is done using the adapted versions[4] of SBX crossover and polynomial mutation operators with the same parameters as are used for continuous variables, except $\eta_c = 3.0$ is used. While all other EMO- and IP2-related parameters are retained, a small adaptation is used in the *offspring's progression* module. After an offspring is advanced (assuming all variables to be continuous), the variables are adjusted to their nearest allowable values.

Both, NSGA-II and NSGA-II-IP2-RF are used to solve this problem, with 16 runs each, and 2500 generations for each run. From Fig. 9, the following can be observed.

[3]The original problem definition had 11 continuous and 13 discrete variables. The continuous variables are converted into discrete, by allowing them to take 1000 equi-spaced values between their respective bounds.

[4]The implementation is taken from https://pymoo.org/operators/

TABLE III
MEDIAN HV (WITH p-VALUE) OF NSGA-III AND NSGA-III-IP2-RF ON FOUR REAL-WORLD PROBLEMS. THE BEST PERFORMING FRAMEWORK IS MARKED IN BOLD

Problem	M	t	NSGA-III	NSGA-III-IP2-RF	p-value
IMPACT	3	100	0.489489	**0.491259**	4.18E-02
CRASH	3	100	0.496360	**0.500835**	1.04E-03
WATER	5	100	0.426957	**0.427958**	7.45E-03
GAA	10	500	0.421847	**0.431046**	2.58E-02

1) The median HV for NSGA-II-IP2-RF is consistently better than that of the original NSGA-II.
2) The median HV achieved by NSGA-II at termination (horizontal black line in Fig. 9(a)) can be achieved by NSGA-II-IP2-RF in just 1410 generations, implying that $\sim$ 44% solution evaluations were saved by NSGA-II-IP2-RF.
3) The final solutions obtained by NSGA-II-IP2-RF are better, compared to NSGA-II, as evident from Fig. 9(b).

E. Other Real-World Problems

Here, four other real-world optimization problems, such as automotive design for side-impact problem [50], automotive frontal crashworthiness design problem [51], water resource problem [52], and the 10-objective General Aviation Aircraft problem [53], are investigated using the IP2 operator. For ease of reference, these problems are referred to as "IMPACT," "CRASH," "WATER," and "GAA," respectively. Their descriptions can be found in their respective references.

All of these problems are solved using NSGA-III and NSGA-III-IP2-RF, while retaining the same EMO- and IP2-related parameters as discussed in Section V. Table III shows: 1) the number of objectives (M); 2) the generation t at which the algorithms are terminated; 3) the median HV for NSGA-III and NSGA-III-IP2-RF; and 4) the p-values using the Wilcoxon ranksum test. It can be observed from Table III that NSGA-III-IP2-RF outperforms NSGA-III (in terms of HV) on every problem. These results on real-world problems are further discussed in Section S11-C of S.D.

VIII. CHALLENGES TO THE UTILITY OF IP2 OPERATOR

The above results on several test and real-world problems have endorsed the potential of the IP2 operator toward performance enhancement of the EMO/EMaO algorithms. However, there are also some inherent limitations, given which the IP2 operator may cease to induce improvements, as articulated with respect to the possible scenarios, below:

1) where the base EMO/EMaO algorithm offers a good PF-approximation, by itself: here, the induced improvements could only be in terms of savings in computational run time to achieve a similar acceptable performance metric value. A run-time analysis of EMO-IP2 and base EMO algorithms (in Section S10 of S.D.) has revealed that savings in computational run time may not be possible if the nature of the given problem is such that the time required for one solution evaluation $\mathcal{T}_{\text{SE}}$ is less than a fraction of the RF training time $\mathcal{T}_{\text{RF}}$: $\mathcal{T}_{\text{SE}} < \kappa\mathcal{T}_{\text{RF}}$, where $\kappa = (1-\rho/\rho)(1/[Nt_{\text{freq}}])$, and ρ is the fraction of generations saved by IP2 operator. Notably, this challenge may manifest frequently in test problems, since $\mathcal{T}_{\text{SE}} \ll \mathcal{T}_{\text{RF}}$ is highly likely. However, it may be reasonable to expect that in real-world problems, where each solution evaluation is computationally expensive, $\mathcal{T}_{\text{SE}} > \kappa\mathcal{T}_{\text{RF}}$ (as illustrated in Section S10 of S.D.), thereby justifying IP2's utility;
2) where the base EMO/EMaO algorithm gets stuck locally and fails to offer a good PF-approximation, by itself: here the potential challenges for the IP2 operator link to the fact that its efficacy is dependent on the performance of the underlying EMO/EMaO algorithm, since the directional improvements are learnt from the past solutions of the same run. In this premise, the IP2 operator may fail to induce improvements, depending on the two subscenarios highlighted as follows.
   a) *EMO/EMaO Population Is Diverse Enough but Fails to Converge:* In such a scenario, the transitions in the variable-space may not be significant enough to allow the RF model to capture the pertinent directional improvements. Despite the provision of jutting to alleviate such a challenge (1), the IP2 operator may at times fail to induce improvements since the apriori chosen jutting-parameter (η) may not always be suitable, for varying combinations of problems and EMO/EMaO algorithm and tuning of η may be necessary.
   b) *EMO/EMaO Population Loses Its Diversity:* In such a scenario, given the extrapolatory limitations of the RF method, and the fact that there is no explicit provision in the IP2 operator to counter lack of diversity, it may fail to induce improvements (detailed in [9]).

   While such challenges exist in principle, the experimental results testify that the degree to which they have manifested on the considered test and real-world problems with varying characteristics, is marginal.

IX. Conclusion

In this article, we have proposed an ML-assisted *enhanced IP* operator, named IP2. It *learns* the pertinent directional improvements in the variable space, from the previous generations, and utilizes this learning to help *advance* some of the current offspring solutions. This entails several modifications in each constitutive module of an earlier proposed IP operator, including: 1) generation of a training dataset that balances the convergence-diversity tradeoff more effectively; 2) learning of the patterns in the training dataset more efficiently, reducing the computational time involved; and 3) the manner and extent to which the trained ML model is utilized, to help the offspring advance more effectively.

The efficacy of the proposed IP2 operator has been tested through its integration with NSGA-II, NSGA-III, MOEA/D, and MaOEA-IGD; comparison with the earlier proposed IP operator on a range of multiobjective problems; extended investigations on many-objective problems ranging up to ten objectives; and analysis on five real-world problems. Evidently, the IP2 operator bears immense potential for performance improvement of EMO/EMaO algorithms, particularly on problems characterized by multimodality, bias, intervariable linkages, and discrete variables, that are otherwise hard to tackle. For many-objective problems, this article has shed light on how the reference-point generation method ought to balance the numbers of boundary and interior points. Toward it, the efficacy of a recently proposed s-energy method [46] is an interesting finding.

While this article has helped identify some of the challenges to the utility of the IP2 operator, it has also been instrumental in identifying several potential research question and directions, including: a more robust criterion for choosing the jutting parameter; extension of the proposed operator for tackling single-objective problems which largely remain unaddressed from the *innovization* perspective; more extensive investigations of discrete- and mixed-variable problems to identify the fuller potential of the *progress* operator; more detailed investigations on the effect of distribution of reference points on the quality of ML *learning* and subsequent progression of offspring; and whether - for a given many-objective problem with redundancy, intermittent ML-training and subsequent progression of offspring can help the underlying EMaO evolve to a lower-dimensional PF, without explicitly resorting to objective-reduction techniques.

The authors believe that this article has only scratched the surface of what may turn out to be one of the dominant research themes of ML-assisted evolutionary optimization. Addressing of some of the above questions will definitely strengthen the prospects for more common adoption of this approach, while tackling real-world problems.

Acknowledgment

The work presented in this article has been developed using the python-based pymoo platform [54].

References

[1] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, U.K.: Wiley, 2001.
[2] C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Boston, MA, USA: Kluwer, 2002.
[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[4] 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 Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[5] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part II: Handling constraints and extending to an adaptive approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, Aug. 2014.

[6] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[7] Y. Du, J.-Q. Li, C. Luo, and L. L. Meng, "A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations," *Swarm Evol. Comput.*, vol. 62, Apr. 2021, Art. no. 100861.

[8] M. S. R. Martins *et al.*, "Analysis of bayesian network learning techniques for a hybrid multi-objective bayesian estimation of distribution algorithm: A case study on MNK landscape," *J. Heuristics*, vol. 27, pp. 549–573, Feb. 2021.

[9] S. Mittal, D. K. Saxena, K. Deb, and E. D. Goodman, "A learning-based *innovized* progress operator for faster convergence in evolutionary multi-objective optimization," *ACM Trans. Evol. Learn. Optim.*, vol. 2, no. 1, pp. 1–29, Nov. 2021. [Online]. Available: https://doi.org/10.1145/3474059

[10] A. Gaur and K. Deb, "Effect of size and order of variables in rules for multi-objective repair-based *innovization* procedure," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2017, pp. 2177–2184.

[11] A. Ghosh, E. Goodman, K. Deb, R. Averill, and A. Diaz, "A large-scale bi-objective optimization of solid rocket motors using *innovization*," in *Proc. IEEE Cong. Evol. Comput. (CEC)*, 2020, pp. 1–8.

[12] S. Mittal, D. K. Saxena, and K. Deb, "Learning-based multi-objective optimization through ANN–Assisted online innovization," in *Proc. Genet. Evol. Comput. Conf. Companion*, New York, NY, USA, 2020, pp. 171–172. [Online]. Available: https://doi.org/10.1145/3377929.3389925

[13] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[14] M. W. Ahmad, M. Mourshed, and Y. Rezgui, "Trees vs neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption," *Energy Build.*, vol. 147, pp. 77–89, Jul. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778816313937

[15] Y. Sun, G. G. Yen, and Z. Yi, "IGD indicator-based evolutionary algorithm for many-objective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 173–187, Apr. 2019.

[16] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000. [Online]. Available: https://doi.org/10.1162/106365600568202

[17] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization* (Advanced Information and Knowledge Processing), A. Abraham, L. Jain, and R. Goldberg, Eds. London, U.K.: Springer, 2005, pp. 105–145. [Online]. Available: https://doi.org/10.1007/1–84628-137–7_6

[18] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.

[19] R. Cheng *et al.*, "A benchmark test suite for evolutionary many-objective optimization," *Complex Intell. Syst.*, vol. 3, pp. 67–81, Mar. 2017.

[20] K. Deb, S. Mittal, D. K. Saxena, and E. D. Goodman, "Embedding a repair operator in evolutionary single and multi-objective algorithms–An exploitation-exploration perspective," in *Evolutionary Multi-Criterion Optimization*, H. Ishibuchi *et al.*, Eds. Cham: Springer, 2021, pp. 89–101.

[21] M. Pelikan, D. Goldberg, and F. Lobo, "A survey of optimization by building and using probabilistic models," *Comput. Optim. Appl.*, vol. 21, no. 1, pp. 5–20, 2002.

[22] L. Li *et al.*, "A robust hybrid approach based on estimation of distribution algorithm and support vector machine for hunting candidate disease genes," *Sci. World J.*, vol. 2013, p. 7, Feb. 2013.

[23] Z. Zhou, Z. Wang, T. Pang, J. Wei, and Z. Chen, "A competition-cooperation evolutionary algorithm with bidirectional multi-population local search and local hypervolume-based strategy for multi-objective optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2021, pp. 153–160.

[24] H. G. Koçer and S. A. Uymaz, "A novel local search method for LSGO with golden ratio and dynamic search step," *Soft Comput.*, vol. 25, no. 3, pp. 2115–2130, 2021.

[25] R. Mallipeddi and M. Lee, "Surrogate model assisted ensemble differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–8.

[26] F. Li, L. Gao, W. Shen, X. Cai, and S. Huang, "A surrogate-assisted offspring generation method for expensive multi-objective optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2020, pp. 1–8.

[27] T. Nakane, X. Lu, and C. Zhang, "A search history-driven offspring generation method for the real-coded genetic algorithm," *Comput. Intell. Neurosci.*, vol. 2020, Sep. 2020, Art. no. 8835852.

[28] C. He, R. Cheng, and D. Yazdani, "Adaptive offspring generation for evolutionary large-scale multiobjective optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Jul. 10, 2020, doi: 10.1109/TSMC.2020.3003926.

[29] S. Qin, C. Sun, Y. Jin, Y. Tan, and J. Fieldsend, "Large-scale evolutionary multiobjective optimization assisted by directed sampling," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 724–738, Aug. 2021.

[30] R. Wang *et al.*, "PCA-assisted reproduction for continuous multi-objective optimization with complicated pareto optimal set," *Swarm Evol. Comput.*, vol. 60, Feb. 2021, Art. no. 100795.

[31] Y. Liu, J. Liu, and Y. Jin, "Surrogate-assisted multipopulation particle swarm optimizer for high-dimensional expensive optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Aug. 12, 2021, doi: 10.1109/TSMC.2021.3102298.

[32] X. Wang, Y. Jin, S. Schmitt, M. Olhofer, and R. Allmendinger, "Transfer learning based surrogate assisted evolutionary bi-objective optimization for objectives with different evaluation times," *Knowl. Based Syst.*, vol. 227, Sep. 2021, Art. no. 107190. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121004524

[33] L. Ma *et al.*, "Learning to optimize: Reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system," *IEEE Trans. Cybern.*, early access, Jul. 14, 2021, doi: 10.1109/TCYB.2021.3086501.

[34] L. Pan, L. Li, R. Cheng, C. He, and K. C. Tan, "Manifold learning-inspired mating restriction for evolutionary multiobjective optimization with complicated pareto sets," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3325–3337, Jun. 2021.

[35] L. Li, C. He, R. Cheng, and L. Pan, "Manifold learning inspired mating restriction for evolutionary constrained multiobjective optimization," in *Evolutionary Multi-Criterion Optimization*, H. Ishibuchi *et al.*, Eds. Cham, Switzerland: Springer, 2021, pp. 296–307.

[36] K. Deb, "Unveiling innovative design principles by means of multiple conflicting objectives," *Eng. Optim.*, vol. 35, no. 5, pp. 445–470, 2003.

[37] K. Deb and A. Srinivasan, "Innovization: Innovating design principles through optimization," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, New York, NY, USA, 2006, pp. 1629–1636.

[38] L. He, H. Ishibuchi, A. Trivedi, and D. Srinivasan, "Dynamic normalization in MOEA/D for multiobjective optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2020, pp. 1–8.

[39] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization," in *Multiple Criteria Decision Making Theory and Applications*, G. Fandel and T. Gal, Eds. Berlin, Germany: Springer, 1980, pp. 468–486.

[40] N. Padhye, K. Deb, and P. Mittal, "Boundary handling approaches in particle swarm optimization," in *Proc. 7th Int. Conf. Bio-Inspired Comput. Theories Appl. (BIC-TA)*, vol. 201, 2013, pp. 287–298.

[41] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "How to specify a reference point in hypervolume calculation for fair performance comparison," *Evol. Comput.*, vol. 26, no. 3, pp. 411–440, Sep. 2018. [Online]. Available: https://doi.org/10.1162/evco_a_00226

[42] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945. [Online]. Available: http://www.jstor.org/stable/3001968

[43] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *J. Amer. Stat. Assoc.*, vol. 47, no. 260, pp. 583–621, 1952. [Online]. Available: http://www.jstor.org/stable/2280779

[44] H. Abdi, "Bonferroni and Šidák corrections for multiple comparisons," in *Encyclopedia of Measurement and Statistics*. Thousand Oaks, CA, USA: SAGE, 2007, pp. 103–107. [Online]. Available: https://ci.nii.ac.jp/naid/20001381227/en/

[45] I. Das and J. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, 1998.

[46] J. Blank, K. Deb, Y. Dhebar, S. Bandaru, and H. Seada, "Generating well-spaced points on a unit simplex for evolutionary many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 1, pp. 48–60, Feb. 2021.

[47] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences* 2nd ed. London, U.K.: Routledge, 1988. [Online]. Available: https://doi.org/10.4324/9780203771587
[48] P. Jain and A. M. Agogino, "Theory of design: An optimization perspective," *Mech. Mach. Theory*, vol. 25, no. 3, pp. 287–303, 1990.
[49] D. Saxena, J. A. Duro, A. Tiwari, K. Deb, and Q. Zhang, "Objective reduction in many-objective optimization: Linear and nonlinear algorithms," *IEEE Trans. Evol. Comput.*, vol. 77, no. 1, pp. 77–99, Feb. 2013.
[50] L. Gu, R. J. Yang, C. H. Tho, L. Makowski, O. Faruque, and Y. Li, "Optimization and robustness for crashworthiness of side impact," *Int. J. Veh. Design*, vol. 26, no. 4, pp. 348–360, 2001.
[51] X. Liao, Q. Li, W. Zhang, and X. Yang, "Multiobjective optimization for crash safety design of vehicle using stepwise regression model," *Struct. Multidiscipl. Optim.*, vol. 35, no. 6, pp. 561–569, 2008.
[52] K. Musselman and J. Talavage, "A tradeoff cut approach to multiple objective optimization," *Oper. Res.*, vol. 28, no. 6, pp. 1424–1435, 1980.
[53] T. Simpson, J. Allen, W. Chen, and F. Mistree, "Conceptual design of a family of products through the use of the robust concept exploration method," in *Proc. 6th AIAA/USAF/NASA/ISSMO Symp. Multidiscipl. Anal. Optim.*, vol. 2, 1996, pp. 1535–1545.
[54] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020.

Sukrit Mittal received the bachelor's degree in mechanical engineering from the Indian Institute of Technology Roorkee, Roorkee, India, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Mechanical and Industrial Engineering.

He had worked with the Mahindra Research Valley, Chennai, India, till 2018. He was a Visiting Scholar for five months with the Michigan State University, East Lansing, MI, USA, in 2020. He is actively pursuing research in AI-assisted optimization, focussing both on interpretable and noninterpretable learning. His research interests include multiobjective optimization, evolutionary computation, and evolutionary design.

Dhish Kumar Saxena received the bachelor's degree in mechanical engineering, the master's degree in solid mechanics and design, and the Ph.D. degree in evolutionary many-objective optimization from the Indian Institute of Technology Kanpur, India, in 1997, 1999, and 2009, respectively.

He worked with the Cranfield University, Cranfield, U.K., and Bath University, Bath, U.K., from 2008 to 2012. He is currently an Associate Professor with the Department of Mechanical and Industrial Engineering, and Mehta Family School of Data Science and Artificial Intelligence, Indian Institute of Technology Roorkee, Roorkee, India. At a fundamental level, his research has focused on objective reduction; constrained reduction; termination criterion; integration of machine learning techniques toward many-objective decision support, and performance enhancement of multi- and many-objective evolutionary algorithms. At an applied level, his focus has been on demonstrating the utility of evolutionary computation and mathematical optimization on a wide range of real world: airline crew scheduling, vehicle routing, engineering design, business process, and multicriterion decision making problems.

Kalyanmoy Deb (Fellow, IEEE) received the bachelor's degree in mechanical engineering from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 1985, and the master's and Ph.D. degrees from the University of Alabama, Tuscaloosa, AL, USA, in 1989 and 1991, respectively.

He is an University Distinguished Professor and a Koenig Endowed Chair Professor with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, USA. His research interests are in evolutionary optimization and their application in multicriterion optimization, modeling, and machine learning. He has published over 575 research papers with Google Scholar citation of over 190 000 with H-index 127.

Prof. Deb was awarded the IEEE Evolutionary Computation Pioneer Award for his sustained work in EMO, the Infosys Prize, the TWAS Prize in Engineering Sciences, the CajAstur Mamdani Prize, the Distinguished Alumni Award from IIT Kharagpur, the Edgeworth-Pareto Award, the Bhatnagar Prize in Engineering Sciences, and the Bessel Research Award from Germany. He is a Fellow of ASME.

Erik D. Goodman received the bachelor's and master's degrees from Michigan State University, East Lansing, MI, USA, in 1966 and 1968, respectively, and the Ph.D. degree from the University of Michigan, Ann Arbor, MI, USA, in 1972.

He is a PI and an Executive Director of the BEACON Center for the Study of Evolution in Action, an NSF Science and Technology Center headquartered at Michigan State University, funded by NSF form 2010 to 2020, and now continuing with funding from MSU. BEACON has a dynamic research program and extensive education and outreach programs, and includes evolutionary biologists as well as computer scientists/engineers studying evolutionary computation (for search and optimization) and evolution of digital organisms. He is a Professor of Electrical and Computer Engineering, Mechanical Engineering, and Computer Science and Engineering. He was a co-founder and VP Technology, Red Cedar Technology, Inc., (currently a division of Siemens), which developed design optimization software that has become a best-selling system in industry.

Prof. Goodman was named the Michigan Distinguished Professor of the Year 2009 and received the MSU Distinguished Faculty Award in 2011. He was an elected Chair of the Executive Board from 2003 to 2005 and a Senior Fellow of International Society for Genetic and Evolutionary Computation; then was a Founding Chair of the ACM SIG on Genetic and Evolutionary Computation 2005. His current personal research is on evolutionary algorithms for optimization of heterogeneous propellant grains for solid-fuel rockets and on evolutionary approaches to neural architecture search.