Investigating Innovized Progress Operators with Different Machine Learning Methods

Drishti Bhasin[1], Sajag Swami[1], Sarthak Sharma[1], Saumya Sah[1], Dhish Kumar Saxena[1 [0000−0001−7809−7744]], and Kalyanmoy Deb[2 [0000−0001−7402−9939]]

[1] Indian Institute of Technology Roorkee, India
{drishti_b,sajag_s,sarthak_s,saumya_s,dhish.saxena}@me.iitr.ac.in
, BEACON Center, Michigan State University, East Lansing, MI USA
kdeb@egr.msu.edu

COIN Report Number 2022012

Abstract. Recent studies have demonstrated that the performance of Reference-vector (RV) based Evolutionary Multi- and Many-objective Optimization algorithms could be improved, through intervention of Machine Learning (ML) methods. These studies have shown as to how *learning* efficient search directions from the intermittent generations' solutions, could be utilized to create pro-convergence and pro-diversity offspring, leading to better convergence and diversity, respectively. The entailing steps of data-set preparation, training of ML models, and utilization of these models, have been encapsulated as *Innovized Progress* operators, namely, IP2 (for convergence improvement) and IP3 (for diversity improvement). Evidently, the focus in these studies has been on proof-of-the-concept, and no exploratory analysis has been done to investigate, *if* and *how drastically* the operators' performance may be impacted, if their underlying ML methods (Random Forest for IP2, and kNN for IP3) are varied. This paper seeks to bridge this gap, through an exploratory analysis for both IP2 and IP3, based on eight different ML methods, tested against an exhaustive test suite comprising of seven multi-objective and 32 many-objective test instances. While the results broadly endorse the robustness of the existing IP2 and IP3 operators, they also reveal interesting tradeoffs across different ML methods, in terms of the Hypervolume (HV) metric and corresponding run-time. Notably, within the gambit of the considered test suite and different ML methods adopted, kNN emerges as a winner for both IP2 and IP3, based on conjunct consideration of HV metric and run-time.

Keywords: Multi-objective Optimization · Many-objective Optimization · Machine Learning assisted Optimization · *Innovized Progress*.

1 Introduction

Reference vector (RV) based Evolutionary Multi- and Many-objective Optimization algorithms [5, 10, 7, 17, 1, 16], hereafter denoted as RV-EMâOAs, are an im-

portant class of algorithms, that seek to find a set of well-distributed Pareto optimal solutions. RV-EMâOAs rely on the use of different dominance principles for convergence, within an RV-based framework which facilitates diversity [16].

Recent studies have shown that RV-EMâOAs are quite conducive for integration with Machine Learning (ML) methods. For instance, it has been demonstrated in [12] that mapping of *inter*-generational solutions *along* the respective RVs in the objective (F) space, could facilitate *learning* of efficient search directions in the variable (X) space, that could be utilized to create pro-convergence offspring. Similarly, [13] has demonstrated that mapping of *intra*-generational solutions *across* different RVs in the F space, could facilitate *learning* of efficient search directions in the X space, that can facilitate pro-diversity offspring.

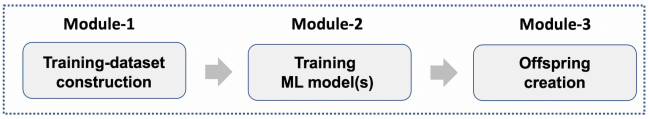


Fig. 1: The constitutive modules for the existing IP2 and IP3 operators

Notably, the above studies rely on three constitutive modules (Figure 1), that include, training-dataset construction, training of ML models, and utilization of these models for offspring creation. These modules when encapsulated for creation of pro-convergence and pro-diversity offspring have been referred to as IP2 [12] and IP3 [13] operators, respectively. It may further be noted that:

- the ML model underlying the IP2 operator is Random Forest, whereas the IP3 operator relies on the use of kNN.
- the IP2 and IP3 operators are not invoked in every generation of an RV-EMâOA run. Instead, their invocation along intermittent generations is determined through a frequency parameter which is determined on-the-fly.
- whenever IP2 is invoked, it is used to create 50% pro-convergence offspring, and the rest are created using the natural variation operators (crossover and mutation). [12] bears evidence that creation of such pro-convergence offspring along intermittent generations, eventually leads to improved convergence.
- wherever IP3 is invoked, it is used to create 50% pro-diversity offspring, and the rest are created using the natural variation operators. [13] bears evidence that creation of such pro-diversity offspring along intermittent generations, eventually leads to improved diversity.
- the hallmark of the ML-based IP2 and IP3 operators is that an RV-EMâOA integrated with either of these operators, *does not* necessitate any additional solution evaluations, compared to the base RV-EMâOA (completely relying on offspring produced by natural variation operators).

While the focus in these studies has been on proof-of-the-concept, no exploratory analysis has been performed yet, to analyse *if* and *how drastically* the

performance of the IP2 and IP3 operators may be impacted, if their underlying ML methods (Module 2, Figure 1) are changed. This paper identifies this gap, and conducts a performance analysis of both IP2 and IP3 operators over an exhaustive test suite, by varying the underlying ML methods.

The remaining paper is organized as follows. Section 2 highlights the ML methods chosen to serve as alternatives to those being used in the existing IP2 and IP3 operators. The experimental settings are highlighted in Section 3, following which the experimental results are presented in Section 4. The paper concludes with Section 5.

2 Alternative ML methods for IP2 and IP3 operators

This paper considers eight different ML methods, covering *Linear and Non-linear Modeling*, *Boosting*, and *Trees*, to investigate their suitability for the IP2 and IP3 operators. The chosen methods are highlighted below:

- from the family of linear methods - standard Linear Regression, Ridge Regression and Elastic Net Regression are included. The Linear Regression based on the least squares approach is used as the base model, but it can also be fitted in other ways, such as by minimizing the *lack of fit* through a penalized version of the least squares cost function as in Ridge Regression (L2-norm penalty) and Lasso (L1-norm penalty). Since Lasso is known to suffer where there are correlated features [9], it has not been considered. Hence, Ridge Regression, and Elastic Net Regression[14] based on a linear combination of the L2-norm and L1-norm penalties, are included.
- from the family of trees - Extra Trees Regressor and Random Forests are included, owing to their ability to efficiently handle complex, non-linear, high-dimensional data, with a lower tendency for *overfitting* [8]. Notably, Random Forests aggregate the results from many decision trees, each generated from a bootstrap sample of the data. Here, at each node, one feature is selected to split on, from a random subset of all features. While in the case of Random Forests, the optimum split is chosen, Extra Trees do it randomly.
- from the family of boosting algorithms - XGBoost is included. It is an implementation of gradient boosting that is scalable and optimized for execution speed and model performance. This is an ensemble technique, where each new model added sequentially learns from the previous models' errors [2].
- other non-linear methods such as k-Nearest Neighbours Regression and Support Vector Regression, are included. The former method identifies the k nearest inputs of the test instance in the original training dataset and returns the average of their respective targets [4]. The latter method seeks to find the hyperplane with the maximum number of points that lies within a threshold distance from the boundary line (unlike other Regression models that try to minimize the error between the real and predicted value).

For ease of reference in the subsequent sections (including Tables and Figures), the chosen ML methods have been abbreviated, as follows: Linear Regression

(LR), Ridge Regression (Ridge), Elastic Net Regression (ENet), Extra Trees Regressor (ExTree) , Random Forests (RF), XGBoost (XGB), k-Nearest Neighbours (kNN), and Support Vector Regression (SVR).

3 Experimental Settings

This section presents the experimental settings for the used - test suite; base RV-EMâOA; and performance indicator.

3.1 Test Suite and the base RV-EMâOA

The considered test suite covers problems with a wide range of characteristics, including, bias, multi-modality, variable-linkages, and different PF shapes (convex, concave, mixed, inverted and disconnected). It includes the following:

- Multi-objective problems: these include: (a) $\tilde{\text{Z}}$DT [11] that are variants of ZDT [18] with modified $g(X)$-functions, leading to the PO solutions at $x_k = 0.5$, for $k = 2, \ldots, 30$, and (b) L1/L2 [11] with variables $n_{\text{var}} = 10$.
- Many-objective problems: these include: (a) DTLZ [6] with distance variables $k = 20$ and $M = 5$, 10, and (b) MaF [3] with $k = 20$ and $M = 5$, 10.

In this paper, NSGA-III has been used as the base RV-EMâOA, for which the SBX crossover ($p_c = 0.9$ and $\eta_c = 20$) and polynomial mutation ($p_m = 1/n_{\text{var}}$ and $\eta_m = 20$) are used. NSGA-III when integrated with the IP2 and IP3 operators, is referred to as NSGA-III-IP2 and NSGA-III-IP3, respectively.

3.2 Performance Indicator

The performance of NSGA-III-IP2 is to be compared amidst its eight variants corresponding to eight different ML methods (identified in Section 2). Since RF is the base method for the existing IP2 operator, NSGA-III-IP2-RF can be distinguished from the other NSGA-III-IP2-ML variants. Similarly NSGA-III-IP3-kNN can be distinguished from the remaining NSGA-III-IP3-ML variants. Since in this paper, NSGA-III is the sole RV-EMâOA, its name can be avoided for brevity, and the task in this paper translates to comparing the performance of: (a) IP2-RF with all other IP2-ML, and (b) IP3-kNN with all other IP3-ML, for which the Hypervolume (HV) measures are used. In that:

- the used Reference Points are given by $R_{1\times M} = [1 + \frac{1}{p}, \ldots, 1 + \frac{1}{p}]$, where p is the number of gaps set for the RV generating Das-Dennis method [13].
- first IP2-RF is run for 21 different seeds, and termination of each such run at say t_{TM} generations is governed by the stabilization tracking algorithm [15] with parameter settings, given by $\psi_{\text{TM}} \equiv \{3, 50\}$ [13]. Then the average of all 21 t_{TM} generations are computed, say $\hat{t}_{\text{TM}}$. Subsequently, each IP2-ML is run for 21 seeds until termination at $\hat{t}_{\text{TM}}$. Finally, the 21 HV measures available for IP2-RF are subjected to Wilcoxon ranksum test (for statistical significance) with a p-value of 0.05, against the 21 HV measures available

for *each* IP2-ML. This test only infers if the difference between IP2-RF and any IP2-ML is statistically insignificant (denoted by =). If not, then their respective median values are directly compared, and the better/worse performing method is denoted by a +/− sign (as in Tables 1 and 2). Similar procedure is adopted for comparison of IP3-kNN with each IP3-ML.

Table 1: HV-based comparison of IP2 with eight different ML methods, where RF is the base for pairwise comparisons. The symbols "+", "=" or "–" indicate whether the corresponding ML method is statistically better, equivalent, or worse, than RF. Here, $\hat{t}_{\mathrm{TM}}$ denotes the average of the termination generations for 21 runs of the base RF.

Problem	M	$\hat{t}_{\mathrm{TM}}$	kNN	ExTree	SVR	ENet	Ridge	LR	XGB	**RF**
L1	2	1058	0.520038+	0.510975=	0.490729=	0.487142–	0.492145=	0.488027=	0.505394=	0.506494
L2	2	1114	0.673404+	0.673393+	0.641642–	0.639171–	0.642941–	0.642229–	0.649486–	0.658637
ZDT1	2	1187	0.681860=	0.681859=	0.681860=	0.681860=	0.681860=	0.681860=	0.681860=	0.681860
ZDT2	2	1289	0.348794=	0.348794=	0.348794=	0.348794=	0.348794=	0.348794=	0.348794=	0.348794
ZDT3	2	1013	1.068370=	1.068457=	1.068402=	1.068368=	1.068490=	1.068559=	1.068454=	1.068502
ZDT4	2	1770	0.681860=	0.681860=	0.681860=	0.681860=	0.681860=	0.681859=	0.681859=	0.681860
ZDT6	2	1779	0.326889+	0.321376=	0.327059+	0.317623=	0.319643=	0.323048=	0.317781=	0.320871
Total (+/=/–)			3/4/0	1/6/0	1/5/1	0/5/2	0/6/1	0/6/1	0/6/1	of 7 probs.
DTLZ1	5	1206	2.485633=	2.486744=	2.486437=	2.486802=	2.486921=	2.486951=	2.486755=	2.479388
	10	2076	17.757704=	17.757704=	17.757704=	17.757707=	17.757711+	17.757708=	17.75771+	17.757703
DTLZ2	5	903	2.172478=	2.172430=	2.172397=	2.172605=	2.172648=	2.17262=	2.172606=	2.172439
	10	797	17.667911=	17.668915=	17.668607=	17.668416=	17.670702+	17.670357+	17.670246+	17.668409
DTLZ3	5	1196	2.111525=	2.110038=	2.086116=	2.103797=	2.102734=	2.109105=	2.083618=	0.975718
	10	1914	17.665819=	17.663740=	17.661898–	17.659894–	17.667285=	17.666322=	17.667097=	17.667027
DTLZ4	5	901	2.173324=	2.173321=	2.173259=	2.173090=	2.173194=	2.173259=	2.173285=	2.173402
	10	873	17.679661–	17.679996=	17.679292–	17.677214–	17.679034–	17.678961–	17.679912=	17.679992
MaF1	5	852	0.060541=	0.060107=	0.059505=	0.059808=	0.060784=	0.059723=	0.060170=	0.061855
	10	938	0.001951=	0.001928=	0.001968=	0.001973=	0.002021+	0.002039=	0.001882–	0.001984
MaF2	5	362	1.002621=	0.998740=	1.002171=	0.999238=	1.003607=	1.001332=	1.000969=	1.000973
	10	492	7.823111=	7.836465=	7.814538=	7.829009=	7.850499+	7.810496=	7.837603=	7.808070
MaF3	5	2509	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320
	10	1912	17.757727+	17.757727+	17.757727+	17.757727+	17.757727+	17.757727+	17.757727+	0
MaF4	5	646	2.475006=	2.474046=	2.483075+	2.472548=	2.466242=	2.477218=	2.480569=	2.462381
	10	1106	17.481657=	17.487127=	17.487090=	17.488453=	17.483530=	17.484489=	17.476629–	17.482483
MaF5	5	1117	2.487940=	2.487937=	2.487936=	2.487937–	2.487939=	2.487938=	2.487937=	2.487940
	10	1180	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727
MaF7	5	1095	0.968628=	0.970901=	0.969541=	0.969176=	0.969783=	0.972155=	0.970013=	0.971900
	10	746	7.634033=	7.626016=	7.620365=	7.579781=	7.637727=	7.631401=	7.619364=	7.611451
MaF8	5	1361	0.000504=	0.000456=	0.000491=	0.000495=	0.000475=	0.000506=	0.000481=	0.000509
	10	1041	0.000087=	0.000087=	0.000082=	0.000082=	0.000081=	0.000080=	0.000084=	0.000085
MaF9	5	2996	0.025761=	0.027336=	0.026762=	0.027222=	0.026980=	0.026736=	0.028677=	0.027412
	10	689	0.000303=	0.000282=	0.000317=	0.000324=	0.000322=	0.000319=	0.000332+	0.000296
MaF10	5	909	0.933306=	0.901095=	0.901880=	0.896384=	0.889783=	0.926323=	0.891255=	0.930609
	10	753	6.298122=	6.102056=	6.318167=	6.310587=	6.247213=	6.255246=	6.269511=	6.225195
MaF11	5	1006	2.452302+	2.442642=	2.449517+	2.437614=	2.421567–	2.420182–	2.448075+	2.439245
	10	1066	17.482878=	17.484652=	17.642898+	17.590201+	17.552494+	17.534068=	17.512623=	17.418106
MaF12	5	539	1.790848=	1.791929=	1.790169=	1.836252=	1.808378=	1.817218=	1.793968=	1.785985
	10	478	13.241563=	13.288255=	13.235779=	13.19385=	13.303345=	13.313089=	13.346762=	13.366043
MaF13	5	871	0.218946=	0.218760=	0.225984=	0.233687+	0.216365=	0.223561=	0.221912=	0.220197
	10	921	0.164408=	0.162449=	0.161884=	0.155932=	0.095460–	0.121185–	0.151681=	0.189759
Total (+/=/–)			2/29/1	1/31/0	4/26/2	3/26/3	6/23/3	2/27/3	5/25/2	of 32 probs.

Table 2: HV-based comparison of IP3 with eight different ML methods, where kNN is the base for pairwise comparisons. The symbols "+", "=" or "–" indicate whether the corresponding ML method is statistically better, equivalent, or worse, than kNN. Here, $\hat{t}_{\text{TM}}$ denotes the average of the termination generations for 21 runs of the base kNN.

Problem	M	$\hat{t}_{\text{TM}}$	RF	ExTree	SVR	ENet	Ridge	LR	XGB	**kNN**
L1	2	1036	0.529465–	0.535453=	0.524300–	0.541379=	0.581979+	0.531304=	0.507512–	0.557609
L2	2	1157	0.661312–	0.658540–	0.659964–	0.659488–	0.661387–	0.664115+	0.660775–	0.662832
ZDT1	2	1200	0.681860=	0.681860=	0.681860=	0.681860=	0.681860=	0.681861=	0.681859=	0.681860
ZDT2	2	1263	0.348794=	0.348794=	0.348794=	0.348794=	0.348794=	0.348794=	0.348794=	0.348794
ZDT3	2	1003	1.068499=	1.068395=	1.068444=	1.068469=	1.068446=	1.068464=	1.068577+	1.068348
ZDT4	2	1755	0.681860=	0.681860=	0.681860=	0.681860=	0.681860=	0.681860=	0.681860=	0.681860
ZDT6	2	1812	0.332970=	0.330529=	0.327835–	0.328804=	0.336573=	0.328397–	0.336123=	0.333947
Total (+/=/–) ⟶			0/5/2	0/6/1	0/4/3	0/6/1	1/5/1	1/5/1	1/4/2	of 7 probs.
DTLZ1	5	1402	2.486749=	2.486892=	2.486839=	2.486717=	2.486795=	2.486833=	2.486964=	2.486885
	10	2023	17.757703=	17.757702=	17.757693=	17.757704=	17.757690=	17.757684=	17.757683=	17.757696
DTLZ2	5	866	2.171945=	2.172083+	2.171872=	2.172031+	2.172102+	2.172082+	2.172134+	2.171782
	10	757	17.664176=	17.664258=	17.664655=	17.664885=	17.664505=	17.664448=	17.664356=	17.664525
DTLZ3	5	1198	2.083931=	2.079724=	2.049543–	2.057957–	2.033053–	2.055357–	2.099911=	2.09651
	10	1851	17.659759=	17.660024=	17.658199–	17.658355=	17.656276–	17.659139=	17.663084=	17.660877
DTLZ4	5	915	2.173070–	2.173172=	2.173212=	2.173206=	2.173125=	2.173185=	2.173211=	2.173235
	10	800	17.676593–	17.67686=	17.67699=	17.676891=	17.676909=	17.676767=	17.677003=	17.676834
MaF1	5	853	0.059759=	0.059340=	0.059473=	0.059903=	0.059967=	0.059876=	0.05956=	0.060380
	10	1059	0.002074+	0.002155+	0.001959=	0.001957=	0.002004=	0.002149+	0.002159+	0.002000
MaF2	5	361	0.999336=	0.997776=	0.999999=	1.000740+	1.002444+	0.998881=	0.999796=	0.995748
	10	484	7.877194=	7.865008=	7.850738=	7.883851+	7.827113=	7.833383=	7.827931=	7.830099
MaF3	5	2416	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320
	10	3658	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727
MaF4	5	649	2.480958=	2.469060=	2.482613=	2.481608=	2.475904=	2.479125=	2.482338=	2.477745
	10	1094	17.529054=	17.534092=	17.541146=	17.497915–	17.565631=	17.535149=	17.542442=	17.557395
MaF5	5	1113	2.487942=	2.487944=	2.487942=	2.487946=	2.487946=	2.487943=	2.487947=	2.487944
	10	1197	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727
MaF7	5	1074	0.968845=	0.972045=	0.967674=	0.969739=	0.968382=	0.968788=	0.969065=	0.969353
	10	734	7.598982=	7.578182=	7.590612=	7.585287=	7.615941=	7.594704=	7.592734=	7.584186
MaF8	5	1361	0.000551=	0.000477=	0.000426–	0.000450–	0.000410-	0.000470=	0.000449–	0.000495
	10	1046	0.000065=	0.000067=	0.000076+	0.000077+	0.000074+	0.000069=	0.000076+	0.000068
MaF9	5	1820	0.028026=	0.028181=	0.025498–	0.026383–	0.027629=	0.028588=	0.026612–	0.028363
	10	725	0.000065=	0.000080=	0.000099=	0.000056=	0.000035=	0.000061=	0.000072=	0.000080
MaF10	5	1069	0.964888=	0.957266=	0.967990+	0.965795=	0.968337+	0.959917=	0.964241+	0.961056
	10	783	6.060485=	5.912519=	5.959022=	6.296865+	5.875247=	5.879183=	6.002479=	5.908347
MaF11	5	960	2.448849=	2.449593=	2.450035=	2.449182=	2.449345=	2.449839=	2.450457=	2.446909
	10	1231	17.561243=	17.571667=	17.567886=	17.551676=	17.550847=	17.558191=	17.549476=	17.557456
MaF12	5	546	1.782436=	1.784627=	1.785285=	1.782553=	1.783997=	1.78398=	1.785365=	1.784192
	10	481	13.118814=	13.118862=	13.130149=	13.165706=	13.122503=	13.077448=	13.243644+	13.041259
MaF13	5	911	0.246188=	0.246621=	0.237909=	0.242349=	0.236794=	0.238086=	0.241078=	0.236727
	10	904	0.237637=	0.236821=	0.230929=	0.192100=	0.225026=	0.229397=	0.235592=	0.226286
Total (+/=/–) ⟶			1/29/2	2/30/0	2/26/4	5/23/4	4/25/3	2/29/1	5/25/2	of 32 probs.

4 Experimental Results

This section presents the comparative performance of: (a) seven IP2-ML variants versus IP2-RF (Table 1), and (b) seven IP3-ML variants versus IP3-kNN (Table 2), across multi- and many-objectives test instances. The dominant trend in both Tables 1 and 2, is that *even if the ML methods underlying the IP2 and IP3 operators are changed, the performance largely remains statistically equivalent (with a few exceptions, discussed later).* This indicates that *the original propositions of IP2 and IP3 are reasonably robust, and their performance may*

neither drastically deteriorate nor improve with a change in the underlying ML method (Module 2, Figure 1). Hence, future attempts to strengthen the Innovized Progress operators may need to focus on improving the other modules, namely, training-dataset generation and offspring creation.

Table 3: Run-time comparison for IP2 and IP3 operators, with eight different ML methods each. Here, RF and kNN are the base methods for the pairwise comparisons within IP2 and IP3, respectively. The entries in the format H/L indicates the number of times an ML method has a Higher/Lower run-time compared to the base method.

		RF	ExTree	SVR	ENet	Ridge	LR	XGB	kNN
IP2	Multi-objective	–	1/6	0/7	1/6	0/7	1/6	5/2	1/6
	Many-objective		6/26	5/27	4/28	4/28	8/24	14/18	8/24
IP3	Multi-objective	5/2	4/3	6/1	3/4	5/2	5/2	7/0	–
	Many-objective	27/5	12/20	17/15	9/23	14/18	10/22	29/3	

For further insights into the results reported in Tables 1 and 2, the notion of *Hypervolume factor* is proposed for each ML method in conjunction with the multi-objective and many-objective test suite. For instance, in Table 1, intersection of Column 3 and shaded Row 9 with the $+/=/-$ entries as 3/4/0, suggests that for the multi-objective test suite comprising of seven problem instances, kNN performed statistically better in three, equivalent in four, and worse in none. In general, for both Tables 1 and 2, if the performance $(+/=/-)$ of any alternative ML method vis-à-vis the base method, for a multi or many-objective problem suite, is given by $B/E/W$, then it is proposed that the *Hypervolume factor* be defined as below.

$$\textit{Hypervolume factor} = \frac{B - W}{B + E + W}. \tag{1}$$

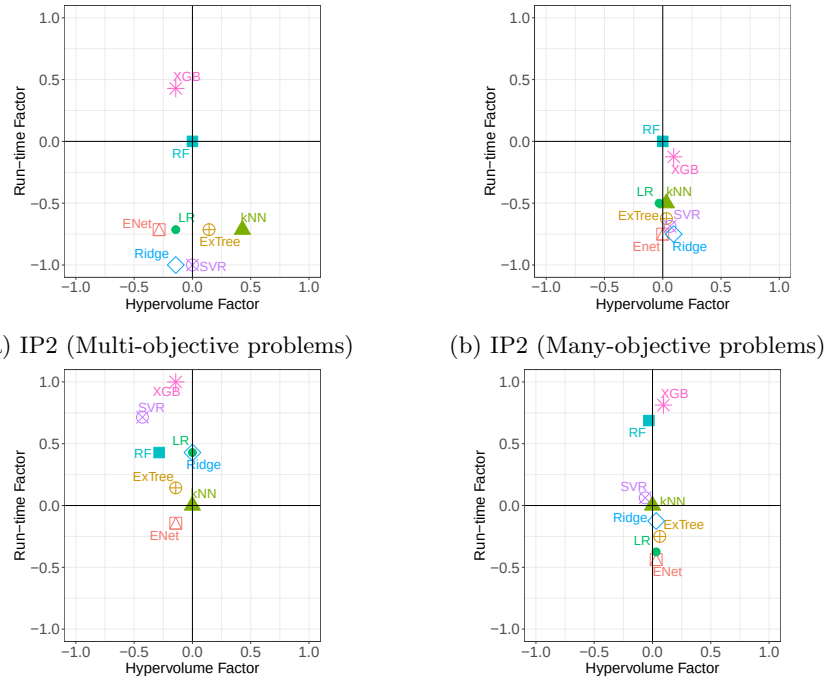
Hence, for any alternative ML method and a specific problem suite: a positive/negative *Hypervolume factor* would represent the percentage instances, where it is *relatively* better/worse than the base method.

Furthermore, the notion of *Hypervolume factor* is extended to computation of *Run-time factor*. As a precursor, it may be noted that the run-time for only the seed underlying the median HV run for each ML method was recorded[2]. Their relative summary formatted as H/L in Table 3, indicates the number of times an ML method has a Higher/Lower run-time compared to the base method. Given this, it is proposed that the *Run-time factor* be defined as below.

$$\textit{Run-time factor} = \frac{H - L}{H + L}. \tag{2}$$

[2] *For this paper, the 21 seed runs were executed in parallel to save the overall run-time, given which the exact run-time for each seed was not traceable. Hence, for run-time estimate, only the seed corresponding to the median hypervolume was executed again.*

Hence, for any alternative ML method and a specific problem suite: a positive/negative *Run-time factor* would represent the percentage instances, where it is *relatively* worse/better than the base method in terms of run-time.



(a) IP2 (Multi-objective problems)

(b) IP2 (Many-objective problems)

(c) IP3 (Multi-objective problems)

(d) IP3 (Many-objective problems)

Fig. 2: Plots comparing HV factor and Run-time factor across the multi-objective and many-objective problem suite for IP2 and IP3 operators.

In the wake of the above, the performance of different ML methods at the level of multi- and many-objective problem-suite is captured in Figure 2. Understandably, RF being the base method for IP2, occupies the origin in Figures 2a and 2b. Similarly, kNN being the base method for IP3, occupies the origin in Figures 2c and 2d. Importantly, if any alternative ML method was to dominate the base method in case of IP2 or IP3, in terms of both HV and run-time, then it should occupy the fourth quadrant. *However, such occurrences are quite rare, as highlighted below*:

- For the IP2 operator applied to multi-objective suite: kNN and ExTree outperform the base RF, and seem to offer reasonably better HV in reasonably lower run-time.

- For the IP2 operator applied to many-objective suite: XGB, kNN, ExTree, SVR, and Ridge seem to offer only a marginally better HV than the base RF, but in reducing order of run-time.
- For the IP3 operator applied to multi-objective suite: the base kNN seems to be the best choice, as it outperforms all other alternative ML methods.
- For the IP3 operator applied to many-objective suite: Ridge, ExTree, LR and ENet offer insignificantly better HV, but in reducing order of run-time.

Overall, if one winner has to be picked across all scenarios, then kNN can be inferred as the one.

The presented results also endorse the known characteristics of some the considered ML methods, as highlighted below. It can be observed that the considered linear ML models, namely LR, Ridge, and ENet have a comparable performance, particularly in terms of the HV measures. Within the family of trees, RF has a higher run-time than ExTree. This is consistent with the expectation, since RF chooses the optimal split, while ExTree relies on random split, saving some time. Overall, the boosting algorithm, namely, XGB is seen to have a higher run-time compared to the other ML methods. This could be attributed to the fact that during each split finding process, XGB iterates over all entries in the input data, making the process slower.

5 Conclusion

This paper sought to analyze, as to how sensitive the existing *Innovized Progress* operators (IP2 for convergence-improvement and IP3 for diversity-improvement) are to the choice of the underlying ML method. In that, the key concern was to investigate if by changing the underlying ML methods, the performance of these operators may drastically deteriorate, or if it could be significantly improved. Exhaustive experiments based on eight ML methods, tested against seven multi-objective and 32 many-objective test instances, suggest that the performance of IP2 could be marginally improved by replacing its base ML method, namely, Random forests, with kNN. However, on the whole, both the existing IP2 and IP3 operators are reasonably robust, and not too sensitive to the choice of underlying ML method. Besides the above inference, the results in this paper also endorsed some of the known characteristics of the considered ML methods. Finally, if one ML method is to be recommended for use, within the gambit of the existing IP2 and IP3 operators; considered test suite; and chosen ML methods, then kNN could be considered as the best performing method. Overall, this paper presents a systematic methodology to investigate the tradeoff associated with different ML methods in terms of their potential for performance enhancement of Evolutionary Multi- and Many-objective Optimization algorithms vis-à-vis the associated computational cost. It is hoped that this shall pave way for similar investigations in other existing ML-based enhancements, such as surrogate-modeling methods.

Acknowledgements. The authors wish to acknowledge Government of India, for supporting this research through an Indo-US SPARC project (code:P66). The authors also wish to thank Sukrit Mittal for his support through this work.

References

1. Chen, L., Liu, H.L., Tan, K.C., Cheung, Y.M., Wang, Y.: Evolutionary many-objective algorithm using decomposition-based dominance relationship. IEEE Transactions on Cybernetics **49**(12), 4129–4139 (2019). https://doi.org/10.1109/TCYB.2018.2859171
2. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. pp. 785–794 (08 2016). https://doi.org/10.1145/2939672.2939785
3. Cheng, R., Li, M., Tian, Y., Zhang, X., Yang, S., Jin, Y., Yao, X.: A benchmark test suite for evolutionary many-objective optimization. Complex & Intelligent Systems **3**, 67–81 (2017). https://doi.org/10.1007/s40747-017-0039-7
4. Cover, T.: Estimation by the nearest neighbor rule. IEEE Transactions on Information Theory **14**(1), 50–55 (1968). https://doi.org/10.1109/TIT.1968.1054098
5. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. Evolutionary Computation, IEEE Transactions on **18**(4), 577–601 (Aug 2014). https://doi.org/10.1109/TEVC.2013.2281535
6. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Evolutionary Multiobjective Optimization: Theoretical Advances and Applications. pp. 105–145. Springer London, London (2005), https://doi.org/10.1007/1-84628-137-7_6
7. Elarbi, M., Bechikh, S., Gupta, A., Ben Said, L., Ong, Y.: A new decomposition-based nsga-ii for many-objective optimization. IEEE Transactions on Systems, Man, and Cybernetics: Systems **48**(7), 1191–1210 (2018). https://doi.org/10.1109/TSMC.2017.2654301
8. Goldstein, B.A., Polley, E.C., Briggs, F.B.S.: Random forests for genetic association studies. Statistical Applications in Genetics and Molecular Biology **10**(1) (2011), https://doi.org/10.2202/1544-6115.1691
9. Hussain, J.N.: High dimensional data challenges in estimating multiple linear regression. Journal of Physics: Conference Series **1591**(1), 012035 (jul 2020), https://doi.org/10.1088/1742-6596/1591/1/012035
10. Liu, J., Wang, Y., Wei, S., Wu, X., Tong, W.: A parameterless penalty rule-based fitness estimation for decomposition-based many-objective optimization evolutionary algorithm. IEEE Access **7**, 81701–81716 (2019). https://doi.org/10.1109/ACCESS.2019.2920698
11. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: A learning-based *Innovized* progress operator for faster convergence in evolutionary multi-objective optimization. ACM Trans. Evol. Learn. Optim. **2**(1) (nov 2021). https://doi.org/10.1145/3474059, https://doi.org/10.1145/3474059
12. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: Enhanced *Innovized* progress operator for evolutionary multi- and many-objective optimization. IEEE Transactions on Evolutionary Computation **26**(5), 961–975 (2022). https://doi.org/10.1109/TEVC.2021.3131952
13. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: A unified *Innovized* progress operator for performance enhancement in evolutionary multi- and many-objective

optimization. Tech. Rep. 2022006, Computational Optimization and Innovation Laboratory, Michigan State University, East Lansing, MI-48824, USA (2022), https://coin-lab.org/content/publications.html
14. Ogutu, J., Schulz-Streeck, T., Piepho, H.P.: Genomic selection using regularized linear regression models: Ridge regression, lasso, elastic net and their extensions. BMC proceedings **6**(2), S10 (05 2012). https://doi.org/10.1186/1753-6561-6-S2-S10
15. Saxena, D.K., Kapoor, S.: On timing the nadir-point estimation and/or termination of reference-based multi- and many-objective evolutionary algorithms. In: Deb, K., Goodman, E., Coello Coello, C.A., Klamroth, K., Miettinen, K., Mostaghim, S., Reed, P. (eds.) Evolutionary Multi-Criterion Optimization. pp. 191–202. Springer International Publishing, Cham (2019)
16. Saxena, D.K., Mittal, S., Kapoor, S., Deb, K.: A localized high-fidelity-dominance based many-objective evolutionary algorithm. IEEE Transactions on Evolutionary Computation pp. 1–1 (2022). https://doi.org/10.1109/TEVC.2022.3188064
17. Yuan, Y., Xu, H., Wang, B., Yao, X.: A new dominance relation-based evolutionary algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation **20**(1), 16–37 (2016). https://doi.org/10.1109/TEVC.2015.2420112
18. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8**(2), 173–195 (2000), https://doi.org/10.1162/106365600568202