

An Interactive Knowledge-based Multi-objective Evolutionary Algorithm Framework for Practical Optimization Problems

Abhiroop Ghosh, *Member, IEEE*, Kalyanmoy Deb *Fellow, IEEE*, Erik Goodman, and Ronald Averill

COIN Report 2023013

Abstract—Experienced users often have useful knowledge and intuition in solving real-world optimization problems. User knowledge can be formulated as inter-variable relationships to assist an optimization algorithm in finding good solutions faster. Such inter-variable interactions can also be automatically learned from high-performing solutions discovered at intermediate iterations in an optimization run – a process called *innovization*. These relations, if vetted by the users, can be enforced among newly generated solutions to steer the optimization algorithm towards practically promising regions in the search space. Challenges arise for large-scale problems where the number of such variable relationships may be high. This paper proposes an interactive knowledge-based evolutionary multi-objective optimization (IK-EMO) framework that extracts hidden variable-wise relationships as knowledge from evolving high-performing solutions, shares them with users to receive feedback, and applies them back to the optimization process to improve its effectiveness. The knowledge extraction process uses a systematic and elegant graph analysis method which scales well with number of variables. The working of the proposed IK-EMO is demonstrated on three large-scale real-world engineering design problems. The simplicity and elegance of the proposed knowledge extraction process and achievement of high-performing solutions quickly indicate the power of the proposed framework. The results presented should motivate further such interaction-based optimization studies for their routine use in practice.

Index Terms—Knowledge extraction, ‘innovization’, interactive optimization, repair, multi-objective optimization.

I. INTRODUCTION

FOR practical multi-objective optimization problems (MOPs), additional knowledge may often be available from the users who have years of knowledge and experience in solving such problems. However, such information is often ignored by researchers while developing an algorithm due to concerns regarding loss of generality. But computational resources for design problems may be limited in time, cost or availability. Thus, in many cases, it may be important to use any available information that may help an optimization algorithm in finding good solutions.

For complex single-objective practical problems, evolutionary algorithms (EAs) with generic recombination and mutation operators [1], [2] may be too slow to lead to high-performing regions of the search space. Good performance of an algorithm in solving benchmark problems such as ZDT [3], DTLZ [4], and WFG [5] does not always translate to good performance

on practical problems. For such cases, creating customized algorithms leveraging additional problem information is necessary. Deb and Myburgh [6] proposed a customized evolutionary algorithm that exploited the linearity of constraint structures to solve a billion-variable resource allocation problem. A micro-genetic algorithm [7] combining range-adaptation and knowledge-based re-initialization was applied to an airfoil optimization problem. Semi-independent variables [8] can be used to handle user-specified monotonic relationships among variables in the form of $x_i \leq x_{i+1} \leq x_{i+2} \leq \dots \leq x_j$. Some techniques for combining EAs with problem knowledge are given in [9]. Domain knowledge can also be semantically annotated and injected into an optimization process [10].

Alternatives to pre-specifying problem information exist, such as cultural algorithms which encode domain knowledge inside a belief space [11]. Self-organizing maps (SOMs) can provide information about important design variable clusters [12]. Recent *innovization* studies [13]–[15] aim to extract additional problem information from high-performing solutions during the optimization process in the form of functional relationships between variables, objectives, and constraints.

Interactive optimization is when the user, referred to as the decision maker (DM), provides guidance during the optimization [16]. Multiple ways to interactively specify information exist, such as aspiration levels [17], importance of individual objectives [18], etc.

Using additional problem information comes with a set of challenges. An effective knowledge representation method needs to be designed which can be used effectively by an optimization algorithm. At the same time, it should also be comprehensible to the user. Validating any user-provided knowledge is necessary since the quality of supplied information may vary. This reduces the possibility of a premature or false convergence. The user may wish to periodically monitor and review optimization progress as well as any learned information. If necessary, the user can also supply information in a collaborative fashion [19]. However, care needs to be taken to ensure that any user feedback does not lead the search process towards sub-optimal solutions. For large-scale problems resulting in a potentially huge rule set, how do we efficiently encode the rule information? How do we ensure that enforcing one rule does not violate one or more of the other rules? How can maximum rule compliance among new solutions be achieved?

This paper aims to address the issues mentioned above by proposing a generic knowledge-based evolutionary multi-

Authors are with Michigan State University, East Lansing, MI 48824, USA, e-mail: {ghoshab1, kdeb, goodman, averillr}@msu.edu (see <https://www.coin-lab.org>).

objective optimization (EMO) framework with user interactivity (IK-EMO) for solving practical constrained multi-objective optimization problems. Users can provide a preference among the learned relationships. The possibility of learned and user-provided knowledge being imperfect is also taken into account and the algorithm can adjust the extent of their influence accordingly. IK-EMO performance is demonstrated on three practical constrained MOPs, and is also compared to three other EMO algorithms.

II. VARIABLE RELATIONSHIPS AS KNOWLEDGE IN AN OPTIMIZATION TASK

Knowledge is a generic term and can be interpreted in many different ways depending on the context. For an optimization task, here, we restrict the definition of knowledge to be additional information provided or extracted about the optimization problem itself. In this paper, we are interested in variable-variable relationships that commonly exist in high-performing solutions of the problem. However, the definition of knowledge can be extended to include the objective and constraint functions too. A practical optimization task minimizes a number of objectives and satisfies a number of constraints, all stated as functions of one or more variables. Thus, understanding the variable-to-variable relationships which are common to feasible solutions (each represented by a variable vector) with small objective values is critically important. A supply of such knowledge *a priori* by the users, in addition to the optimization problem description, or a discovery process of such knowledge from the evolving high-performing optimization solutions, can be directly utilized by the optimization algorithm to speed up its search process. Moreover, if such knowledge is discovered during the optimization process, users will benefit from having this knowledge in addition to the optimal solutions of the problem.

A. Past studies

Innovization is the process of extracting commonalities among Pareto-optimal solutions, first proposed by Deb and Srinivasan [13]. The basic principle of innovization is to generate rules representing inter-variable relationships in simple forms such as power laws ($x_i x_j^b = c$). In [20], the authors have proposed a method which is able to express relationships involving operators like summation (+), difference (−), product (×), etc.

Bandaru and Deb [14] introduced the concept of higher- and lower-level innovization. A genetic programming-based innovization framework was proposed in [20] and was applied on an inventory management problem. An MOEA combined with a local search procedure was employed in [21] to ensure faster convergence. A combination of innovization and data mining approaches were used in [22] to achieve faster convergence. Gaur and Deb [15] proposed an adaptive innovization method that treats the innovization process as a machine learning problem and repairs the solutions directly, based on the learned model. A combination of user guidance and inequality relation-based online innovization [23], [24] was used to solve three practical problems.

B. Structure of rules considered in this study

For an interactive knowledge-based optimization algorithm to work, a standard form of knowledge representation is necessary which is simple enough for users to understand but has enough complexity to capture problem knowledge accurately. Using algebraic expressions or ‘rules’ is one way of representing knowledge and has been extensively used in the ‘innovization’ literature [13], [15]. A rule can take the form of an equality or an inequality, as shown below:

$$\phi(\mathbf{x}) = 0, \quad (1)$$

$$\psi(\mathbf{x}) \leq 0. \quad (2)$$

Any arbitrary form of rules involving many variables from a decision variable vector (\mathbf{x}) and complicated mathematical structures of functions ϕ or ψ may be considered, but such rules would not only be difficult to learn, they would also be difficult to interpret by the user. In this study, we restrict the rules to have simple structures involving a maximum of two variables, as discussed below.

1) *Constant rule*: This type of rule involves only one variable taking a constant value ($x_i = \kappa_i$). In terms of Equation 1, for the i -th variable, the structure of the rule becomes $\phi_i(\mathbf{x}) = x_i - \kappa_i$. This type of rule can occur if multiple high-performing solutions are expected to have in common a fixed value of a specific variable [25].

2) *Power law rule*: Power law rules [13] for two variables x_i and x_j can be represented by Equation 1 as $\phi_{ij}(\mathbf{x}) = x_i x_j^b - c$, where b and c are constants. This form makes power laws versatile enough to encode a wide variety of rules, such as proportionate or inversely proportionate relationships among two variables. Interestingly, an inequality power law using a ψ function can also be implemented, but such a rule may represent a relationship loosely and we do not consider it here.

3) *Equality rule*: This type of rule can express the equality principle of two variables x_i and x_j observed in high-performing solutions. In terms of Equation 1, $\phi_{ij}(\mathbf{x}) = x_i - x_j$ is the rule’s structure.

4) *Inequality rule*: This type of rule can represent relational properties of two variables x_i and x_j as $x_i \leq x_j$ or $x_i \geq x_j$. In terms of Equation 2, $\psi_{ij}(\mathbf{x}) = x_i - x_j$ or $\psi_{ij}(\mathbf{x}) = x_j - x_i$ are the respective rules. For example, the radius of two beams in a truss [23] might be related via this type of rule.

After describing the chosen rule structures, we are now ready to discuss the procedures of extracting such rules from high-performing variable vectors and applying the extracted rules to the optimization algorithm. A summary of their representations and use in our analysis are provided in Table I.

III. PROPOSED INTERACTIVE KNOWLEDGE-BASED IK-EMO FRAMEWORK

In this study, we restrict our discussions to multi-objective optimization problems, so high-performing solutions refer to the entire *non-dominated* (ND) solution set discovered by the optimization algorithm from the start of the run to the current iteration. Figure 1 shows the proposed IK-EMO framework.

The framework starts with a description of the multi-objective optimization problem, as shown in the top-left box in

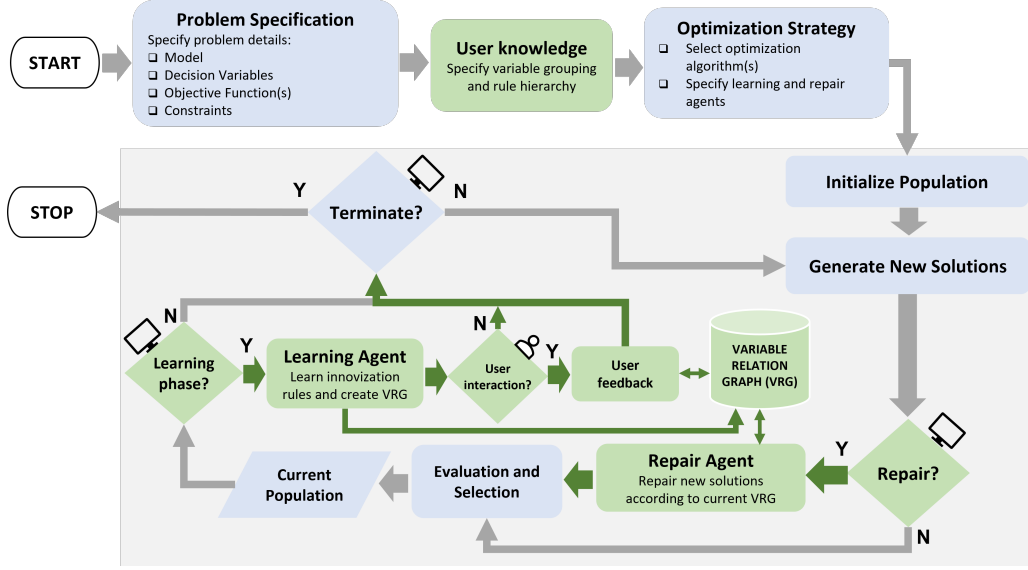


Fig. 1: Interactive knowledge-based EMO framework (IK-EMO) showing user interaction, learning and repair agents. Blue blocks represent a normal EMO. Green blocks represent the components responsible for knowledge extraction and application, as well as user interaction.

the figure. In addition, if any additional problem information is available, that is also specified. The penultimate step before starting the optimization is to select a suitable EMO and methods to algorithmically extract and apply any problem knowledge. The subsequent sections describe the various components in more detail.

A. User knowledge

Before the start of the optimization, the user may provide some initial information which will affect how the framework operates, details of which are given below.

1) *Variable grouping*: For a problem with n variables, there can be $\frac{n(n-1)}{2}$ pairwise variable interactions. For any reasonable-sized problem, such a huge number of meaningful relationships may not exist. In practice, the user may be interested in only a handful of relationships that relate some critical decision variables. In order to reduce the complexity, variables can be divided into different groups G_k for $k = 1, 2, \dots, n_g$. Each group consists of variables that the user thinks are likely to be related. Group information is specified prior to the optimization. If the user does not know how to construct the groups, he/she can put all n variables into a single group. It is then up to the algorithm to figure out which variables are related. Inter-group relationships are not discoverable under this scheme. Variables that are not part of any group are assumed not to be related to other variables. For example, assume there are two variable groups $G_1 = \{2, 3, 5\}$ and $G_2 = \{1, 4, 7\}$ for an 8-variable problem. For G_1 all pairwise combinations (x_2, x_3) , (x_2, x_5) , and (x_3, x_5) will be checked for the existence of any possible relationships. A similar process is repeated for G_2 . Since inter-group relationships are not explored, combinations like (x_3, x_7) will not be considered. Variables x_6 , and x_8 are not part of any group, hence they are assumed not to be related to the other variables in any meaningful way.

2) *Rule hierarchy*: In the proposed framework we consider four types of rules as presented in Section II. The user may be interested in more than one type of rule and have some preferences among them. In that case, the user can define a hierarchy of multiple rule types ranked according to user preference. The existence of a particular rule type for one or more variables is checked rank-wise. For example, if constant rules are ranked 1, followed by power laws (rank 2) and inequalities (rank 3), then the variables in every group will be checked for constant rules first. The variables which do not exhibit constant rules will then be checked for power laws, and so on. For relations having equal ranking, a scoring criterion needs to be used to determine which rule better represents the non-dominated (ND) front and will be used by the algorithm.

B. Learning agent

A learning agent is a procedure used to identify different innovization rules present in the ND solutions in a population. The rules involve a single variable or a pair of variables, as required by a rule's description. Each type of rule (inequality, power law, etc.) requires a different rule satisfaction condition. A score (within $[0,1]$, as presented in Table I) is assigned to each rule to quantify how well the rule represents the ND set. Different learning agents applicable to the rule types covered in Section II are presented below. A summary of the various rules, their scoring procedures and satisfaction conditions are provided in Table I.

1) *Constant rule*: In order to learn constant rules, we have to analyze the values of the variable under consideration for every ND solution and check if one or more of them converge to specific values. Since variables can be of different scales and units, we need a generalized criterion to determine if a variable is taking on a constant value. First, the median (\hat{x}_i) is calculated. The proportion of ND solutions which satisfy

TABLE I: Rule types and the corresponding mathematical representation. \mathcal{X} represents the set of ND solutions. x_i and x_j refer to the i -th and j -th variables, respectively, of a ND solution $\mathbf{x} \in \mathcal{X}$. The corresponding variables in a new solution \mathbf{x}_r to be repaired are labeled as x_{ir} and x_{jr} , respectively. Normalized variables are represented by a hat (\hat{x}_i, \hat{x}_j). Higher ranked rules are preferred while performing repair. The score (s) is a measure of how well \mathcal{X} follows the rule in the representation column. Satisfaction condition dictates whether \mathbf{x}_r follows the respective rule.

Rule type	Representation	Score	Satisfaction condition
Constant	$\phi_i(\mathbf{x}) = x_i - \kappa_i = 0$	$s_{\phi_i} = \frac{ A_i }{ \mathcal{X} }$, where $A_i = \{1 : \forall \mathbf{x} \in \mathcal{X}, x_i - \kappa_i \leq \rho_i\}, \kappa_i = \tilde{x}_i$	$ x_{ir} - \kappa_i \leq \rho_i$
Power law	$\phi_{ij}(\mathbf{x}) = \hat{x}_i \hat{x}_j^b - c = 0$	$s_{\phi_{ij}} = R^2$ score of linear regression shown in Equation 4	$\left(\hat{x}_{jr} - \left(\frac{c}{\hat{x}_{ir}}\right)^{\frac{1}{b}}\right)^2 \leq e_{ij}^{\min}$
Equality	$\phi_{ij}(\mathbf{x}) = x_i - x_j = 0$	$s_{\phi_{ij}} = \frac{ B_{ij} }{ \mathcal{X} }$, where $B_{ij} = \{1 : \forall \mathbf{x} \in \mathcal{X}, x_i - x_j \leq \varepsilon_{ij}\}$	$ x_{ir} - x_{jr} \leq \varepsilon_{ij}$
Inequality (\leq)	$\psi_{ij}(\mathbf{x}) = (x_i - x_j) \leq 0$	$s_{\psi_{ij}} = \frac{ C_{ij} }{ \mathcal{X} }$, where $C_{ij} = \{1 : \forall \mathbf{x} \in \mathcal{X}, x_i \leq x_j\}$	$(x_{ir} - x_{jr}) \leq 0$
Inequality (\geq)	$\psi_{ij}(\mathbf{x}) = (x_j - x_i) \leq 0$	$s_{\psi_{ij}} = \frac{ D_{ij} }{ \mathcal{X} }$, where $D_{ij} = \{1 : \forall \mathbf{x} \in \mathcal{X}, x_i \geq x_j\}$	$(x_{jr} - x_{ir}) \leq 0$

$|x_i - \tilde{x}_i| \leq \rho_i$ is said to be the score (s_{ϕ_i}) of the constant rule $x_i = \kappa_i = \tilde{x}_i$. ρ_i is a small tolerance used for determining whether variable x_i 's value is in the neighborhood of \tilde{x}_i . It must be defined separately for each variable. An alternative option is to normalize the variables and define a singular ρ for the normalized variable space. To check whether a new solution (\mathbf{x}_r) follows $x_{ir} = \kappa_i$, we check whether x_{ir} lies in the neighborhood of κ_i using the condition: $|x_{ir} - \kappa_i| \leq \rho_i$.

2) *Power law rule*: In order to learn power laws ($x_i x_j^b = c$) we use the method proposed in [26] with a modification. Each variable is initially normalized to [1, 2]. A training dataset is created from the ND solution set with the logarithms of normalized variables \hat{x}_i and \hat{x}_j as features, leading to Eqn. 4:

$$\hat{x}_i \hat{x}_j^b = c, \quad (3)$$

$$\Rightarrow \log \hat{x}_i = \beta \log \hat{x}_j + \epsilon, \quad (4)$$

where $\beta = -b$ is the weight and $\epsilon = \log c$ is the intercept. Normalization prevents 0 or negative values from appearing in the logarithm terms. Then we apply ordinary least squares linear regression to the logarithm of \hat{x}_i and \hat{x}_j . Linear regression finds the best-fit line for the training data defined by the parameters β and ϵ . In order to evaluate the quality of the fit, we use the coefficient of determination (R^2) metric. A new solution (\mathbf{x}_r) follows the power law given in Equation 3 if the difference between the actual value (x_{ir} or x_{jr}) and the predicted value (\hat{x}_{ir} or \hat{x}_{jr}) is lower than a pre-defined threshold error (e_{ij}^{\min}). Table I shows the formulation for the satisfaction condition.

3) *Equality rule*: Two variables can be considered equal if $|x_i - x_j| \leq \varepsilon_{ij}$ with ε_{ij} being a tolerance parameter for variable pair x_i and x_j . The proportion of ND solutions following this condition is the score ($s_{\phi_{ij}}$) of the equality rule. The need to define ε_{ij} for every variable pair can be avoided if normalized variables are used.

4) *Inequality rule*: Inequality rules can be of the form $x_i \leq x_j$ or $x_i \geq x_j$. The proportion of ND solutions satisfying either condition is the score of the respective rules.

After the learning agent identifies specific rules from a set of ND solutions, the rules can be used to repair offspring solutions of the next generation. The repair mechanism for each rule is described next.

C. Repair agent

Once the rules are learned from the current ND solutions by the learning agent, the next task is to use these rules to repair the offspring solutions for the next few generations. There are two questions to ponder. First, how many rules should we use in the repair process? Second, how closely should we adhere to each rule while repairing? A small fraction of learned rules may not embed requisite properties present in the ND solutions in offspring solutions. But the usage of too many rules may reduce the effect of each rule. Similarly, a tight adherence to observed rules may encourage premature convergence to a non-optimal solution, while a loose adherence may not pass on properties of ND solutions to the offspring. We propose four different rule usage schemes (10% (RU1) to 100% (RU4)) and three rule adherence schemes (RA1 (tight) to RA3 (loose)) for power law and inequality rules.

1) *Constant rule*: To apply a constant rule $x_i = \kappa_i$ to a particular offspring solution $\mathbf{x}^{(k)}$, the variable $x_i^{(k)}$ is simply set to κ_i , thereby implementing the learned rule from previous ND solutions to the current offspring solutions. Constant rules are always included in the rule set and used with tight adherence.

2) *Power law rule*: For a power law rule $\hat{x}_i \hat{x}_j^b = c$, one variable is selected as the base (independent) variable and the other variable is set according to the rule. For example, for a particular offspring solution $\mathbf{x}^{(k)}$, if $\hat{x}_i^{(k)}$ is selected as the base variable, $\hat{x}_j^{(k)}$ is set as follows: $\hat{x}_j^{(k)} = \left(\frac{c}{\hat{x}_i^{(k)}}\right)^{\frac{1}{b}}$. Despite theoretically being able to represent constant relationships by having $b = 0$, in practice, extremely low values of b can cause the repaired variable $\hat{x}_j^{(k)}$ to have a large value outside the variable range. Hence, in this study, we first check whether a variable follows constant rules, and if it does, then that variable's involvement in a power law rule is ignored.

A repair of a power law rule is followed with three different confidence levels by adjusting to an updated c -value: $\hat{x}_i \hat{x}_j^b = c_r$. PL-RA1 uses $c_r = c$ (tight adherence); PL-RA2 uses $c_r \in \mathcal{N}(c, \sigma_c)$ (medium adherence), and PL-RA3 uses $c_r \in \mathcal{N}(c, 2\sigma_c)$ (loose adherence), where σ_c is the standard deviation of c -values for the power law observed among the ND solutions during learning process. PL-RA1 puts the greatest trust into the learned power law rule, whereas PL-RA3 has the least amount of trust and provides the most flexibility in the repair process.

3) *Inequality and equality rules:* In order to repair an offspring solution $\mathbf{x}^{(k)}$, we have to select one variable ($x_i^{(k)}$) as the base variable and the other ($x_j^{(k)}$) as the dependent variable to be repaired. The generalized inequality repair operation is shown below:

$$x_j^{(k)} = x_i^{(k)} + \nu_{r1}(x_i^U - x_i^{(k)}), \quad \text{for } x_i^{(k)} \leq x_j^{(k)}, \quad (5)$$

$$x_j^{(k)} = \frac{x_i^{(k)} - \nu_{r2}x_i^U}{1 - \nu_{r2}}, \quad \text{for } x_i^{(k)} \geq x_j^{(k)}. \quad (6)$$

Three different rule adherence (RA) schemes are considered. IQ-RA1 uses $\nu_{r1} = \mu_{\nu1}$ and $\nu_{r2} = \mu_{\nu2}$ (tight adherence with no standard deviation), which are computed as the means of ν_1 and ν_2 from ND solutions during the learning process, as follows:

$$\nu_1 = \frac{x_j - x_i}{x_i^U - x_i}, \quad \nu_2 = \frac{x_i - x_j}{x_i^U - x_j}.$$

For IQ-RA2, $\nu_{r1} \in \mathcal{N}(\mu_{\nu1}, \sigma_{\nu1})$ and $\nu_{r2} \in \mathcal{N}(\mu_{\nu2}, \sigma_{\nu2})$ (medium adherence with one standard deviation) are used, where $\sigma_{\nu1}$ and $\sigma_{\nu2}$ are standard deviations of ν_1 and ν_2 , respectively. Both ν_{r1} and ν_{r2} are set to zero, if they come out to be negative. For IQ-RA3, $\nu_{r1}, \nu_{r2} \in U(0, 1)$ (loose adherence with a uniform distribution) are used.

D. Ensemble repair agent

Both power law and inequality/equality rules have three rule adherence options for repair. For a new problem, it is not clear which option will work the best, so we also propose an ensemble approach (PL-RA-E and IQ-RA-E) in which all three options are allowed, but based on the success of each option, more probability is assigned to each. The ensemble method also considers a fourth option in which no repair to an offspring is made. The survival rate (r_s^i) of offspring generated by the i -th repair operator is a measure of its quality. The greater the survival rate of the offspring created by an operator is, the higher is the probability of its being used in subsequent offspring generation. The probability (\hat{p}_r^i) update operation for the i -th operator is presented below:

$$p_r^i(t+1) = \max \left(p_{\min}, \alpha \frac{r_s^i}{\sum_i r_s^i} + (1 - \alpha) \hat{p}_r^i(t) \right), \quad (7)$$

$$\hat{p}_r^i(t+1) = \frac{p_r^i(t+1)}{\sum_i p_r^i(t+1)}, \quad (8)$$

where α is the learning rate, $r_s^i = \frac{n_s^i}{n_{\text{off}}}$, where n_s^i and n_{off} are the number of offspring created by the i -th operator that survive in generation t and the total number of offspring that survive in generation t , respectively. It is possible that at any point during the optimization, no solution generated by one of the repair operators survives. This might cause the corresponding selection probability to go down to zero without any possibility of recovery. To prevent this, in Equation 7, the probability update step ensures that a minimum selection probability (p_{\min}) is always assigned to each repair operator present in the ensemble. Equation 8 normalizes the probability values for each operator so that their total sum is one.

The learning rate (α) determines the rate of change of the repair probabilities. A high α would increase the sensitivity,

and can result in large changes in repair probabilities over a short period of time. A low α exerts a damping effect which causes the probability values to update slowly. Through trial and error, $\alpha = 0.5$ and $p_{\min} = 0.1$ are found to be suitable for the problems of this study.

E. Mixed rule repair agent

A mixed rule repair agent is designed to work on two or more different types of rules. Since multiple rules (for example, an inequality rule and a power law rule) can show up for the same variable pair, a rule hierarchy needs to exist as defined in Section III-A2. Table II shows the rule hierarchical rank used for all the repair agents in this study.

TABLE II: Rule hierarchy by rank for each repair agent.

Repair agent	Rule type	Rank
PL-RA1, PL-RA2, PL-RA3, PL-RA-E	Constant	1
	Power law	2
IQ-RA1, IQ-RA2, IQ-RA3, IQ-RA-E	Constant	1
	Equality	2
	Inequality (\leq)	3
	Inequality (\geq)	3
Mixed (Power law and inequality)	Constant	1
	Power law	2
	Equality	2
	Inequality (\leq)	2
	Inequality (\geq)	2

F. User's ranking of rules

The user forms the basis of the interactivity of the IK-EMO framework. At any point during the optimization, the user has the option to review the optimization results and provide feedback to the optimization algorithm in one or more of the following ways:

- **Rule ranking:** The user may provide a ranking of rules (rank 1 is most preferred) provided by the algorithm. The algorithm will then try to implement the rules in the rank order provided by the user.
- **Rule exclusion:** The user may select to remove certain rules provided by the algorithm, based on their knowledge of the problem.
- **Rule specificity:** The user may specify details for considering a rule further. For example, the user may specify that only variables having a correlation above a specified value should be considered. Another criterion could be to select all rules having a score greater than a threshold as rank 1 and exclude the others.

In this paper, the proposed rule usage schemes (RU1-RU4) can also be considered as artificial users [27] who select a certain percentage of the learned rules every few generations. This systematically illustrates the interactive ability of IK-EMO while showing the effect of different numbers of rules used for repair on the performance.

In the subsequent experiments, it is assumed that the user instantaneously provides their feedback. However, in the real world, this may not be the case. The user may require some finite time to adequately process the results and gather their feedback. Pausing the optimization algorithm during the user's

analysis process can result in losing out on useful function evaluations that could have been completed during this overall allocated time period. An experimental analysis of this issue is provided in the supplementary document.

G. Variable relation graph (VRG)

The possible number of pair-wise relations among n variables is $\frac{n(n-1)}{2}$ or $O(n^2)$. Thus, for a large number of variables, the amount of bookkeeping required to track individual pairwise relations is large. Moreover, the observed relationships should not contradict each other. For example, for inequality rules $x_i \leq x_j$ and $x_j \leq x_k$, the transitive property can be maintained by choosing to repair x_j based on x_i , followed by repairing x_k based on x_j using Equation 5. But repairing both x_j and x_k separately based on x_i can potentially contradict the rule $x_j \leq x_k$. To solve these two challenges, we propose using a graph-based data structure, called variable relation graph (VRG), to encode and track relationships observed between multiple variable pairs. A customized graph-traversal algorithm ensures that all repairs are performed with minimal or no contradictions. In the following sections, steps 1 to 5 show the process of using learning agents to construct a VRG (learning phase). A learning interval (T_L) is defined as the number of generations or function evaluations (FEs) after which a new learning phase begins. Step 6 shows the process of applying the VRG to repair an offspring solution using one or more repair agents (repair phase). A repair interval (T_R) is defined as the number of generations or FEs between any two repair phases.

1) *Create a complete VRG*: A vertex (or node) of a VRG represents a variable and an edge connecting two nodes indicates the existence of a relationship between the corresponding variables. For every group G_k of variables, all pairwise variable combinations are connected by an edge. This will result in a *complete* graph where every pair of vertices is connected by a unique undirected edge. An example with two variable groups ($G_1 = \{1, 2, 3, 6, 8\}$ and $G_2 = \{4, 5, 7, 9, 10\}$) having five variables each is illustrated in Figure 2.

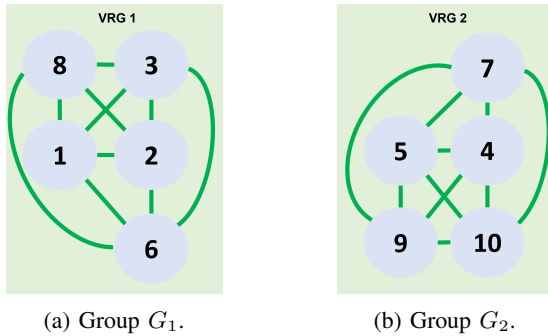


Fig. 2: Ten variables in two non-interacting groups are represented in complete graphs. Each node represents a variable. An edge $i-j$ represents the existence of a relationship between decision variables x_i and x_j .

2) *Rule selection*: In this step, learned rules are used to modify the VRGs according to two selection criteria. First,

all rules having a score (defined in Table I) above a certain threshold (s_{\min}) are considered. Second, they are applied in the order of user's preference ranking. A connection may be removed if it does not satisfy the selection criteria. If a single-variable (constant) rule satisfies the selection criterion, then the corresponding node is removed from the VRG and that rule will be implemented separately. If no two-variable rule involving x_i and x_j satisfies the minimum score criterion, the corresponding VRG edge ($i-j$) is removed. An example is shown in Figure 3, which uses the rule hierarchy for mixed rule repair operators (third row) shown in Table II, except that inequalities are ranked 3 for illustration here. A blue or brown edge represents a power law rule or an inequality rule, respectively. An edge ranking is also assigned based on the rule hierarchy. In this case, edges representing power laws and inequalities will be ranked 1 and 2 by default, unless overruled by the user. Both graphs have a reduced number of edges after the rule selection process is complete. Node 8 in Figure 3a (marked in red) is found to have a constant rule associated with it and hence removed. In Figure 3b, variables (x_5, x_9) and (x_9, x_{10}) are not related by power laws having a score greater than s_{\min} . However, they are found to follow inequality relationships with a score greater than s_{\min} . Hence, they are connected by brown edges. The rest of the edges represent power law rules and are marked by blue. The approach to set the direction of the edges is discussed next.

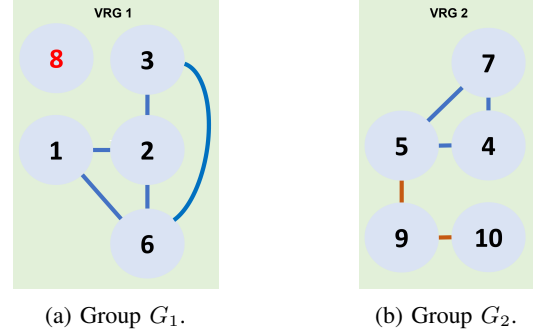


Fig. 3: A blue edge represents that a corresponding power law relation has a score greater than s_{\min} . A brown edge represents a corresponding inequality relation having a high score. If the scores for both types of relations are low, the corresponding edge is removed. If this results in a node having no edges, then it is removed, such as node 8 in Figure 3a.

3) *Create a directed acyclic VRG*: In order to apply a repair agent to the VRG, it needs to be converted to a directed acyclic graph (DAG). This step ensures graph traversal is possible without getting stuck in loops. The members of every group G_k are randomly permuted to create a sequence D_k . If i appears before j in D_k , an undirected edge between nodes i and j is converted to a directed edge from i to j . In the example shown in Figure 4, two random sequences $D_1 = (2, 1, 3, 6)$ and $D_2 = (10, 4, 5, 9, 7)$ are created for groups G_1 and G_2 , respectively. Since node 2 appears before node 1 in D_1 , a blue arrow goes from node 2 to node 1, as shown in the figure. This process is repeated for every population member so as to create diverse VRGs.

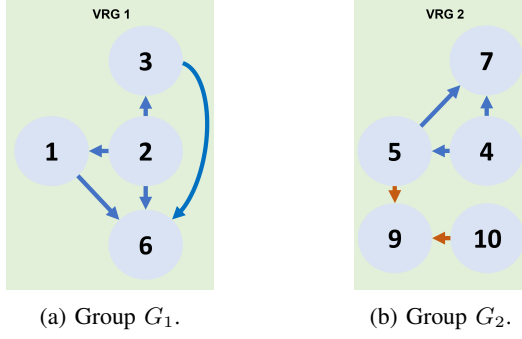


Fig. 4: A random sequence of the nodes is created for each group. An edge $i \rightarrow j$ is created from an undirected edge if i appears before j in the sequence. This creates a directed acyclic variable relation graph.

4) *Transitive reduction*: Next, a transitive reduction [28] is performed on the VRG corresponding to each variable group. For VRGs having both power law and inequality edges, transitive reduction is performed on subgraphs consisting only of the edges of the same type. This step eliminates redundant directed edges between two different rule types. An example of eliminating an arrow from node 2 to node 6 is shown in Figure 5a.

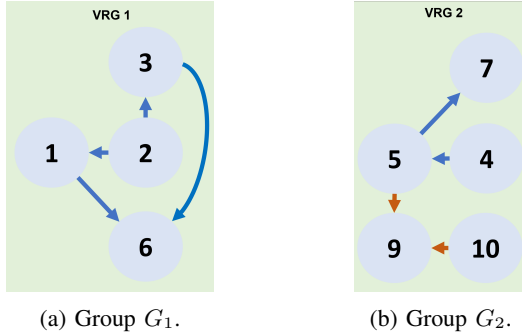


Fig. 5: Transitive reduction is performed to remove redundant directed edges. In Figure 5a, since edges $2 \rightarrow 3 \rightarrow 6$ exists, edge $2 \rightarrow 6$ is considered redundant and is removed.

5) *Modify VRG according to user's feedback*: A user can provide feedback in the form of a ranking, or select only a subset of the available rules. In the former case, the VRG edge rankings are updated to reflect the user's choice. Edges corresponding to the rules discarded by the user are removed. Figure 6a shows an example where the rule involving x_1 and x_6 are ranked 1 (marked by arrows with a red border) and x_2 and x_3 are ranked 2 (marked by arrows with a dark yellow border). The gray edges represent the rules discarded by the user. Figure 6b shows a similar ranking process.

6) *Repair new offspring solutions*: For every new solution, the corresponding VRGs are traversed. From a starting node, the algorithm moves forward via the outgoing edges and repairs the connected nodes recursively in a depth-first fashion. This is repeated for all ranks. Algorithm 1 presents the pseudocode of the repair process. In the pseudocode, the VRG data structure has the attributes Nodes and Edges. The

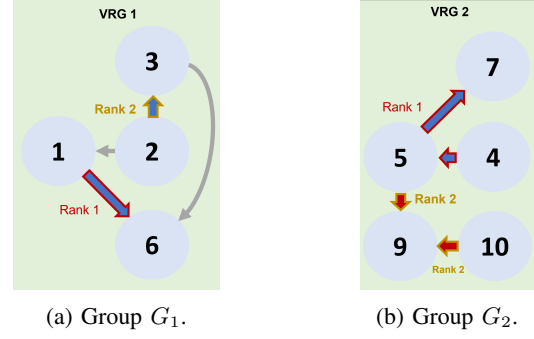


Fig. 6: User can provide feedback by adding, removing, or ranking the rules. Ranking divides an existing variable relation graph into multiple sub-graphs. Edges with a red border represent rank 1 relations. Edges with a dark yellow border represent rank 2 relations. Gray edges represent relations that the user wants to remove.

Edges attribute representing an edge (i, j) has multiple sub-attributes: StartVertex (i in this case), EndVertex (j in this case), EdgeType (rule type and corresponding repair agent), EdgeRank (rank of an edge). A function *TraverseGraph* is used which recursively traverses the VRG from a random start node for a particular rule rank. The function *Repair* called by *TraverseGraph* calls the correct repair agent based on EdgeType.

Algorithm 1 VRG traversal and repair pseudocode.

Require: New solution set (\mathcal{X}_r), variable groups (G), VRGs for every solution and group, rule hierarchy.

Ensure: Repaired solution set \mathcal{X}_r .

```

1: function TRAVERSEGRAPH( $x$ , Graph, CurrentNode, PreviousNode, NodesVisited, CurrentRank)
2:   if CurrentNode in NodesVisited then return
3:   end if
4:   CurrentEdges  $\leftarrow$  Graph.Edges[CurrentNode];
5:   for each outgoing edge ( $e$ ) in CurrentEdges do
6:     NextNode  $\leftarrow$   $e$ .EndVertex;
7:     if NextNode not in NodesVisited then
8:       EdgeType  $\leftarrow$   $e$ .EdgeType;
9:       EdgeRank  $\leftarrow$   $e$ .EdgeRank;
10:      if EdgeRank = CurrentRank then
11:        Repair( $x$ , CurrentNode, NextNode, EdgeType, EdgeRank);
12:      end if
13:      TraverseGraph( $x$ , Graph, NextNode, CurrentNode, NodesVisited);
14:    end if
15:  end for
16:  Add CurrentNode to NodesVisited;
17: end function
18: for each group  $G_k$  in  $G$  do ▷ Repair procedure begins
19:   for each solution  $x$  in  $\mathcal{X}_r$  do
20:     CurrentGraph  $\leftarrow$  VRG assigned to  $x$  for  $G_k$ ;
21:     for CurrentRank = 1, 2, ...,  $n_{ranks}$  do
22:       StartNode  $\leftarrow$  Select the first node having atleast one edge of rank CurrentRank from a random sequence;
23:       TraverseGraph( $x$ , CurrentGraph, StartNode, NULL, [], CurrentRank);
24:     end for
25:   end for
26: end for
```

TABLE III: Number of decision variables and constraints for 39 and 59-segment stepped beams.

n_{seg}	$\sigma_{max}(MPa)$	$\delta_{max}(m)$	Width (b_i)	Height (h_i)	Aspect ratio (a_i)	Variables	Constraints
39	20	0.04	[0.1, 40]	[0.1, 40]	[0.5, 2]	78	41
59	20	0.06	[0.1, 60]	[0.1, 60]	[0.5, 2]	118	61

IV. SIMPLY-SUPPORTED STEPPED BEAM DESIGN

Beam design problems are common in the literature [8], [29] and can be used to benchmark an optimization algorithm. In this paper, we consider a simply-supported stepped beam design with multiple segments having a rectangular cross-section. An example with five segments is shown in Figure 7. A vertical load of 2 kN is applied at the middle of the

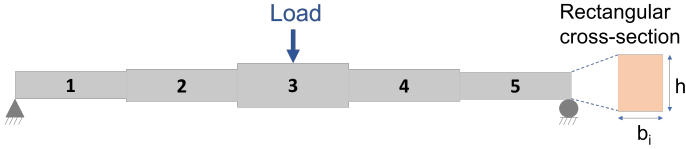


Fig. 7: Simply-supported stepped beam with five segments.

beam. All n_{seg} segments are of equal length. The area of the rectangular cross-section is determined by its width (b_i) and height (h_i) for the i -th segment, where $i \in [1, n_{seg}]$. The volume (V) and maximum deflection (Δ) are to be minimized by finding an optimal width b_i and height h_i of each segment, totalling $2n_{seg}$ variables. The maximum stress $\sigma_i(\mathbf{x})$ of i -th member and deflection $\delta_j(\mathbf{x})$ at j -th node need to be kept below strength of the material σ_{max} and a specified limit δ_{max} , respectively. The aspect ratio (ratio of height to width) of each segment is also restricted within a particular range (in $[a_L, a_U]$), as constraints. The MOP formulation is shown below:

$$\text{Minimize } V(\mathbf{x}) = \sum_{i=1}^{n_{seg}} b_i h_i l_i, \quad (9)$$

$$\text{Minimize } \Delta(\mathbf{x}) = \max_{i=1}^{n_{seg}} \delta_i(\mathbf{x}), \quad (10)$$

$$\text{Subject to } \max_{i=1}^{n_{seg}} \sigma_i(\mathbf{x}) \leq \sigma_{max}, \quad (11)$$

$$\max_{j=1}^{n_{seg}} \delta_j(\mathbf{x}) \leq \delta_{max}, \quad (12)$$

$$a_L \leq a_i \leq a_U, \text{ for } i = 1, \dots, n_{seg}. \quad (13)$$

Here, two cases with 39 and 59 segments are considered. Problem parameters are described in Table III.

A. Experimental settings

NSGA-II [30], a state-of-the-art MOEA, is applied with the proposed IK-EMO procedure to solve both cases. This problem is intended to demonstrate the performance of our proposed algorithm with minimal initial user knowledge. Thus, all variables are put into a single group. IK-EMO is combined separately with each repair agent described in Section III-C. In addition, there are two cases where mixed relationships are used: the first case with PL-RA2 and IQ-RA2, and the second case with PL-RA-E and I-ES. The rule hierarchy is described in Table II.

Four rule usage schemes RU1, RU2, RU3 and RU4 select the top 10%, 20%, 50% and 100% of the learned rules sorted according to their scores. They also act as artificial users with a consistent behavior. Each rule usage scheme is paired with one or more repair agents. Eight cases with a single repair agent are considered: PL-RA1, PL-RA2, PL-RA3, PL-RA-E, IQ-RA1, IQ-RA2, IQ-RA3, IQ-RA-E. Two cases with a combination of repair agents are considered: one with PL-RA2 and IQ-RA2, and the other with PL-RA-E and IQ-RA-E. From Table I, ρ_i is set to be 0.1, ε_{ij} is set as 0.1, and e_{ij}^{min} is set to 0.01. Table IV shows the parameter settings for this problem.

For each combination of a repair agent and user, 20 runs are performed and the Hypervolume (HV) [31] values are recorded at the end of each generation. The Wilcoxon rank-sum test [32] is used to compare the statistical performance of the algorithms tested here with respect to the best performing algorithm for each scenario. As an example, let p_1 and p_2 represent the performance metric values for two algorithms A_1 and A_2 . For each simulation run, p_1 and p_2 exist as paired observations. Here, the null hypothesis states that there is no statistically significant difference between p_1 and p_2 . The hypothesis is tested with 95% significance level and the p -values are recorded. A p -value less than 0.05 means that there is a statistically significant performance difference between A_1 and A_2 . The median number of FEs taken to achieve a target hypervolume (HV^T) is used as a performance metric for the Wilcoxon test. HV^T is set to be the final median HV achieved by base NSGA-II when the run is terminated. In order to make the problem challenging for the proposed approach, a small population size of 40 is used and a maximum number of generations of 500 is set, thereby allowing a maximum computational budget of 20,000 FEs for each run.

TABLE IV: Parameter settings of IK-EMO.

Parameter	Value
Population size	Problem-specific
Maximum generations	Problem-specific
Mutation operator	Polynomial mutation [33]
Mutation probability (p_m) and index (η_m)	$1/n_{var}$, 50
Crossover operator	SBX [1]
Crossover probability (p_c) and index (η_c)	0.9, 30
Minimum rule score, s_{min}	0.7
Learning interval (T_L , in generations)	10
Repair interval (T_R , in generations)	10
α and p_{min} in Equation 7	0.5, 0.1
Rule parameters ρ_i , ε_{ij} , e_{ij}^{min}	Problem-specific

B. Experimental results and discussion

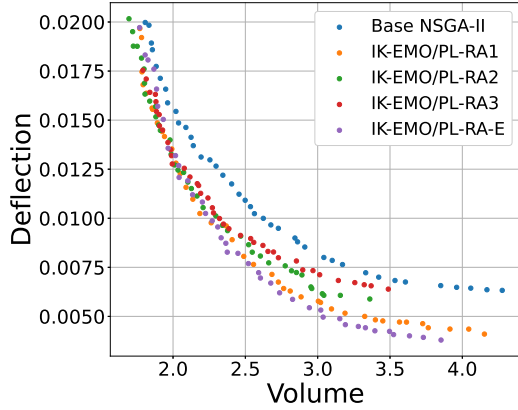
Tables V and VI show the optimization results for the 39 and 59-segment stepped beam problems, respectively. Base NSGA-II results without any rule extraction and repair are shown in the first row. The best performance case in each

TABLE V: FEs required to achieve $HV^T = 0.95$ for 39-segment beams. Best performing algorithm for row is marked in bold. Best performing algorithm in each column is marked by a shaded gray box. Algorithms with statistically similar performance to the best algorithm column-wise are marked in italics. The corresponding Wilcoxon p-values are given in braces.

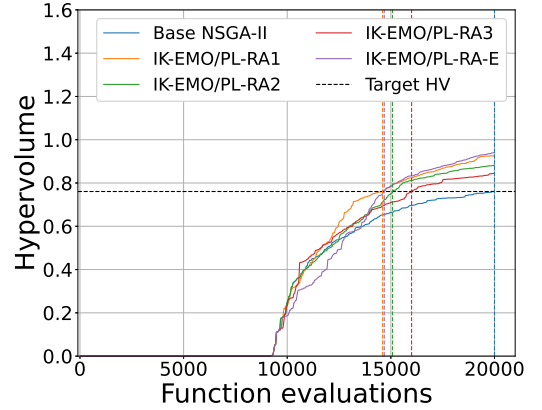
Rule Type	Repair agent	RU1	RU2	RU3	RU4
None	None (base)	20.0k (p=0.0144)	20.0k (p=0.0124)	20.0k (p=0.0255)	20.0k (p=0.0421)
Power law rule	PL-RA1	16.7k \pm 0.5k (p=0.0224)	15.6k \pm 1.1k (p=0.0660)	15.8k \pm 0.7k (p=0.0421)	17.9k \pm 1.2k (p=0.0309)
	PL-RA2	16.1k \pm 1.8k (p=0.0199)	<i>16.0k \pm 2.1k</i> (p=0.0615)	15.5k \pm 1.5k (p=0.0710)	17.3k \pm 1.0k (p=0.0189)
	PL-RA3	19.8k \pm 0.4k (p=0.0011)	19.4k \pm 1.2k (p=0.0405)	19.5k \pm 1.3k (p=0.0300)	19.5k \pm 1.2k (p=0.0405)
	PL-RA-E	14.7k \pm 2.0k	14.6k \pm 0.1k	14.7k \pm 1.9k	15.1k \pm 1.8k
Inequality rule	IQ-RA1	17.6k \pm 0.6k (p=0.0022)	17.5k \pm 1.2k (p=0.0473)	17.7k \pm 0.9k (p=0.0467)	17.8k \pm 1.0k (p=0.0309)
	IQ-RA2	17.2k \pm 1.0k (p=0.0135)	17.3k \pm 1.2k (p=0.0265)	17.3k \pm 0.8k (p=0.0488)	17.7k \pm 1.2k (p=0.0202)
	IQ-RA3	19.8k \pm 1.2k (p=0.0481)	19.6k \pm 1.0k (p=0.0341)	19.5k \pm 1.1k (p=0.0322)	19.7k \pm 1.1k (p=0.0249)
	IQ-RA-E	17.6k \pm 0.8k (p=0.0413)	17.5k \pm 0.9k (p=0.0217)	17.6k \pm 1.2k (p=0.0335)	17.6k \pm 1.5k (p=0.0185)
Mixed rule	PL-RA2+IQ-RA2	15.0k \pm 0.9k (p=0.0511)	14.9k \pm 1.1k (p=0.0993)	15.3k \pm 1.1k (p=0.0686)	15.4k \pm 0.8k (p=0.0720)
	PL-RA-E+IQ-RA-E	14.9k \pm 1.0k (p=0.0921)	14.8k \pm 0.7k (p=0.0766)	15.0k \pm 0.7k (p=0.0718)	15.2k \pm 0.7k (p=0.0512)

TABLE VI: FEs required to achieve $HV^T = 0.76$ for 59-segment beams.

Rule Type	Repair agent	RU1	RU2	RU3	RU4
None	None (base)	20.0k (p=0.0112)	20.0k (p=0.0018)	20.0k (p=0.0011)	20.0k (p=0.0025)
Power law rule	PL-RA1	15.8k \pm 0.5k (p=0.0106)	14.6k \pm 2.2k	15.8k \pm 0.5k (p=0.0365)	16.4k \pm 0.8k (p=0.0381)
	PL-RA2	15.0k \pm 1.6k	<i>15.1k \pm 1.9k</i> (p=0.0640)	14.6k \pm 2.8k (p=0.1010)	15.7k \pm 1.7k (p=0.0512)
	PL-RA3	16.5k \pm 2.0k (p=0.0045)	16.0k \pm 1.5k (p=0.0218)	16.8k \pm 1.4k (p=0.0220)	17.3k \pm 2.6k (p=0.0433)
	PL-RA-E	15.1k \pm 1.1k (p=0.1261)	14.7k \pm 0.3k (p=0.0911)	14.5k \pm 1.0k	15.2k \pm 1.7k
Inequality rule	IQ-RA1	16.7k \pm 4.0k (p=0.0214)	16.4k \pm 2.5k (p=0.0411)	16.9k \pm 2.1k (p=0.0185)	18.5k \pm 2.2k (p=0.0321)
	IQ-RA2	16.9k \pm 4.3k (p=0.0201)	16.0k \pm 2.9k (p=0.0389)	17.2k \pm 3.3k (p=0.0169)	18.0k \pm 2.6k (p=0.0342)
	IQ-RA3	16.5k \pm 4.3k (p=0.0311)	17.4k \pm 3.0k (p=0.0341)	17.7k \pm 3.2k (p=0.0163)	18.2k \pm 2.4k (p=0.0253)
	IQ-RA-E	16.7k \pm 3.3k (p=0.0291)	16.6k \pm 4.4k (p=0.0282)	17.3k \pm 2.4k (p=0.0305)	18.4k \pm 1.9k (p=0.0244)
Mixed rule	PL-RA2+IQ-RA2	15.3k \pm 3.1k (p=0.0585)	14.8k \pm 3.0k (p=0.0910)	14.7k \pm 3.0k (p=0.0723)	15.5k \pm 2.5k (p=0.0699)
	PL-RA-E+IQ-RA-E	15.1k \pm 2.4k (p=0.0923)	14.7k \pm 2.7k (p=0.0623)	14.7k \pm 2.6k (p=0.0720)	15.4k \pm 2.1k (p=0.0511)



(a) ND Front for one run.



(b) HV plot over 20 runs.

Fig. 8: ND fronts and hypervolume plots obtained by IK-EMO with RU2 and power law repair agents for 59-segment stepped beam problem.

row is marked in bold. For every column, the best performing algorithm is marked with a shaded gray box. The Wilcoxon p-values show the relative performance of each algorithm with the column-wise best performance. Algorithms with a statistically similar performance to the column-wise best are shown in italics. The ND front obtained in a particular run using the power law repair operators for RU2 are shown in Figure 8a for the 59-segment case. The corresponding median HV plot over the course of the optimization run are shown in Figure 8b. Similar behaviors are observed for 39-segment case (see supplementary materials).

The results show many interesting observations as stated below.

1) *General observations*: Statistically base NSGA-II does not perform well compared to knowledge-based NSGA-II methods for both 39- and 59-segment problems. A positive aspect of the proposed algorithm is that it is still able to achieve a good performance with significantly low population size. For problems with expensive evaluation functions, this may stay beneficial for saving computational time.

2) *Power law vs inequality rules*: As can be seen from the table, for both 39- and 59-segment problems, the power law repair operators generally perform better than inequality-based

repair operators except for PL-RA3. One possible reason could be the greater versatility of power laws in modeling complex relationships compared to simple inequality rules.

3) *Best performing algorithm for each rule usage scheme:* In the 39-segment case, PL-RA-E is the best performer for RU1 to RU4, with PL-RA-2 and the mixed repair operators offering statistically similar performance for RU1 to RU3. This shows that the ensemble method can be used to get good performance without the need for selecting a proper repair process. For the 59-segment case, PL-RA2 and PL-RA1 are the best performers for RU1 and RU2, respectively. PL-RA-E offers a statistically similar performance in both cases. In the cases of RU3 and RU4, PL-RA-E gives the best performance, with PL-RA2 having a comparable performance. PL-RA3 performs significantly worse than the other PL operators for all the cases. For PL-RA1, adhering closely to the learned power law rules constrains NSGA-II in finding good solutions. PL-RA3 introduces a large amount of variance which is detrimental to the optimization process. A compromise between these two extremes, provided by PL-RA2 or PL-RA-E, is the logical step.

Figure 8 illustrates the results when RU2 is combined with the power law repair operators for both problem cases. The difference in the quality of solutions obtained after 20,000 FEs is prominent in the 59-segment ND front. In the median HV plots it is seen that the FEs required to reach HV^T for PL-RA-E is close to the number needed by the best performing repair agent. Base NSGA-II and the repair operators all give good quality solutions at the end of the run for the 39-segment case. However, for the 59-segment case, base NSGA-II performs significantly worse.

4) *Relative performance of each rule usage scheme:* It can be seen from Tables V and VI that RU2 produces the best performance in 7 out of 10 cases for the 39-segment case, and 6 out of 10 cases for the 59-segment case. This shows that in terms of rule usage, using too few or too many of the learned rules is not effective in improving the algorithm's performance.

5) *Mixed relation repair agents:* The mixed relation repair agents (PL-RA2+IQ-RA2) and (PL-RA-E+IQ-RA-E), have statistically similar performance to the best algorithm for each user and both problem cases. Even though inequality repair operators perform worse than power law repair operators individually, their presence in the mixed repair agents do not hinder the performance, since only the high-performing rules are added to the VRG during creation. The proposed framework is robust enough to give good performance irrespective of the number of repair agents and type of rules.

V. OPTIMAL POWER FLOW PROBLEM

Optimal power flow (OPF) is a common problem in power system engineering with MOEAs being used to solve the problem [34], [35]. The following objective functions are minimized: fuel cost, emissions, voltage deviation, and real power loss. In many cases, one or more of these objectives are considered in the literature, with the rest being kept as constraints. In this study, we consider two objectives: minimizing

fuel cost and reducing fossil fuel emissions. Voltage deviation and power loss are kept as constraints. This version of the OPF problem is also known as the environmental economic dispatch (EED) problem [35].

$$\text{Minimize } C_F(\mathbf{P}_G, \mathbf{V}_G) = \sum_{i=1}^{N_G} (a_i + b_i P_{Gi} + c_i P_{Gi}^2), \quad (14)$$

$$\text{Minimize } C_E(\mathbf{P}_G, \mathbf{V}_G) = \sum_{i=1}^{N_G} (\alpha_i + \beta_i P_{Gi} + \gamma_i P_{Gi}^2 + \zeta_i e^{(\lambda_i P_{Gi})}), \quad (15)$$

$$\begin{aligned} \text{Subject to } & \sum_{i=1}^{N_{bus}} (P_i - P_D - P_L) = 0, \\ & VD_{min} \leq VD \leq VD_{max}, \quad P_{Lmin} \leq P_L \leq P_{Lmax}, \\ & Q_{Gimin} \leq Q_{Gi} \leq Q_{Gimax}, \quad P_{smin} \leq P_s \leq P_{smax}, \\ & V_{smin} \leq V_s \leq V_{smax}, \quad V_{PQimin} \leq V_{PQi} \leq V_{PQimax}, \end{aligned} \quad (16)$$

where C_F is the fuel cost, C_E is the emission cost, N_G is the number of generators, P_{Gi} is the real power output and V_{Gi} is the voltage output of the i^{th} generator, (a_i, b_i, c_i) are the fuel cost coefficients, $(\alpha_i, \beta_i, \gamma_i, \zeta_i, \lambda_i)$ are the emission cost coefficients. VD is the total voltage deviation of all the load buses, P_L is the total real power loss, Q_{Gi} is the reactive power output of the i^{th} generator, P_s is the real power output and V_s is the voltage output of the slack bus, V_{PQi} is the voltage at the i^{th} load/P-Q bus. P_D is the power demand and N_{bus} is the total number of buses. A load flow analysis must be performed to satisfy the equality constraint. We use MATPOWER [36] as the load flow solver. We consider IEEE 118-bus and 300-bus systems in this study.

A. Experimental settings

The bus details, along with the numbers of decision variables and constraints, are given in Table VII. The types of decision variables and their corresponding ranges are given in Table VIII.

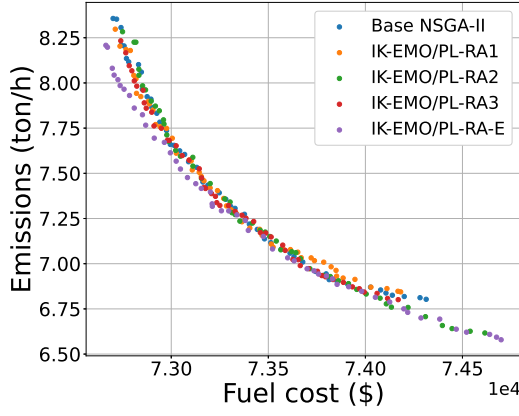
TABLE VII: IEEE bus system specifications.

System	Generators	Transformers	Load bus	Decision variables	Constraints
IEEE 118-bus	54	11	64	115	240
IEEE 300-bus	69	107	231	243	604

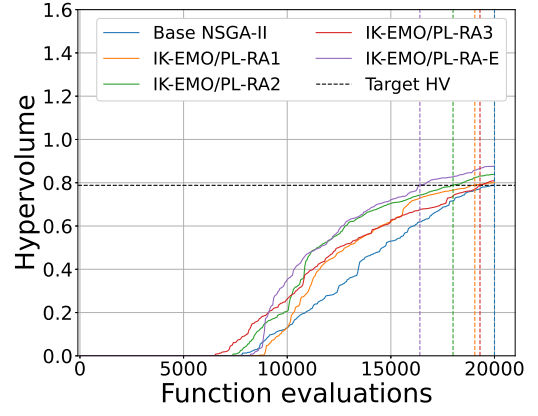
TABLE VIII: OPF decision variable types and ranges.

Variable	Range
Generator power output (P_{Gi})	[30, 100]
Generator voltage (V_{Gi})	[0.95, 1.05]
Transformer tap ratio (T_i)	[0.9, 1.1]

Experimental settings are the same as in the stepped beam problem except that the population size is set to be 50 and the maximum number of generations is set as 400 for both IEEE 118 and 300-bus systems. From Table I, ρ_i and ε_{ij} are set as 1, and e_{ij}^{min} is set to 0.01. Two variable groups are defined and shown in Table IX.



(a) ND Front for one run.



(b) HV plot over 20 runs.

Fig. 9: ND front and hypervolume plots obtained with RU2 and power law repair agents for IEEE 300-bus OPF problem.

TABLE IX: OPF variable groups.

Group	Variable Type	Variable Indices	
		118-bus	300-bus
G_{opf1}	Generator power and voltage	[1-104]	[1-136]
G_{opf2}	Transformer tap ratio	[105-115]	[137-243]

B. Experimental results and discussion

Tables X and XI show the optimization results for the IEEE 118- and 300-bus systems, respectively. The ND front obtained in a single run using four power law repair methods for RU2 are shown in Figure 9a for the IEEE 300-bus system. The corresponding median HV plots over the course of the optimization are shown in Figure 9b. Similar behavior is observed for the 118-bus system (see supplementary document).

1) *General observations*: Base NSGA-II is outperformed by both the power law and inequality-based repair agents. Good performance with low population size is obtainable by IK-EMO, making it the better choice for this problem.

2) *Power law vs inequality rules*: For both problem cases, a power law repair operator is the best performer for each user, as in the stepped beam problem. Inequality-rule-based repair operators in general result in worse performance compared to power-law-based repair operators. This, as in the stepped beam problem, is a result of the power laws being able to more accurately model the inter-variable relationships.

3) *Best performing algorithm for each user*: For both the 118 and 300-bus problems, PL-RA-E performance is the best or statistically similar to the best for all the cases. Thus, an ensemble repair agent is a good choice for automatically selecting the best repair agent according to the situation.

Figure 9 illustrates the results with RU2 combined with the power law repair operators for the IEEE 300-bus case. The difference in the quality of solutions obtained after 20,000 FEs is prominent in the IEEE 300-bus case. In the median HV plots it is seen that the number of FEs required to reach HV^T for PL-RA-E is the least, followed by PL-RA-2.

4) *Relative performance of each rule usage scheme*: It can be seen from Tables X and XI that RU2 produces the best performance in 9 out of 10 cases for the IEEE 118-bus case,

and 8 out of 10 cases for the IEEE 300-bus case. This shows that in terms of rule usage, using too few or too many of the learned rules is detrimental to the optimization performance in general, which is similar to the conclusions made in the stepped beam design problems.

5) *Mixed relation repair agents*: The mixed relation repair agent PL-RA-E+IQ-RA-E, has statistically similar performance to the best algorithm for each user and both problem cases. As in the stepped beam problems, the worse performance of the inequality repair operators does not hinder the performance of the mixed relation operators.

VI. TRUSS DESIGN PROBLEM

Finally, we consider a commonly-used truss design problem involving two objectives, and 1,416 highly nonlinear constraints. The truss has 1,100 members and 316 nodes, making a total of 1,179 variables, making it a large-scale problem. The details of the problem description are provided in the supplementary document.

Experimental settings are similar to those of the previous problems. Population size is set to 100 and the maximum number of generations is set as 10,000. Thus, the total computational budget comes out to be 1 million FEs. From Table I, ρ_i and ε_{ij} are set as 0.1, and e_{ij}^{min} is set to 0.01. Multiple variable groups are defined for this problem based on the relative location and alignment of the beams as shown in Table XII.

A. Experimental results and discussion

Results are presented in Table XIII. Base NSGA-II is outperformed by most repair operators. As in the previous two problems, power-law-based approaches perform better than inequality-based approaches, but the ensemble-based approach performs overall the best with an intermediate use of repair (RU2). More information are put in supplementary document.

VII. SUMMARY OF RESULTS

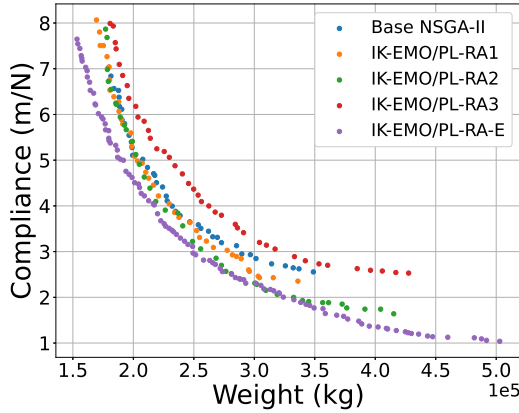
For every problem we have a total of 11 different algorithms including base NSGA-II and 10 repair schemes. For each

TABLE X: FEs required to achieve $HV^T = 0.90$ for IEEE 118-bus system.

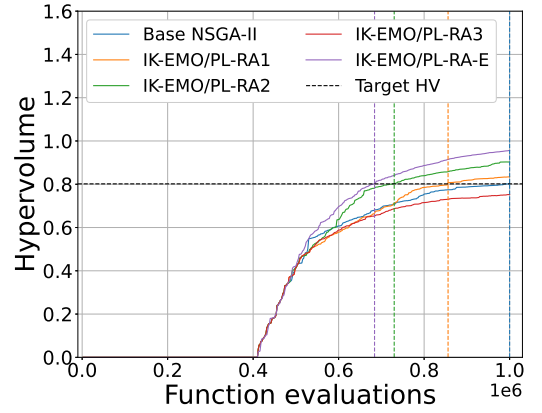
Rule Type	Repair agent	RU1	RU2	RU3	RU4
None	None (base)	20k (p=0.0350)	20k (p=0.0322)	20k (p=0.0310)	20k (p=0.0430)
Power law rule	PL-RA1	17.6k \pm 0.2k (p=0.0101)	17.1k \pm 0.9k (p=0.0417)	17.8k \pm 0.5k (p=0.0119)	18.0k \pm 0.2k (p=0.0313)
	PL-RA2	17.4k \pm 0.4k (p=0.0180)	17.1k \pm 0.4k (p=0.0205)	17.4k \pm 0.2k (p=0.0156)	17.6k \pm 0.3k
	PL-RA3	18.0k \pm 0.5k (p=0.0152)	17.6k \pm 0.7k (p=0.0398)	18.0k \pm 0.4k (p=0.0117)	18.2k \pm 0.6k (p=0.0086)
	PL-RA-E	16.1k \pm 0.4k	15.9k \pm 0.5k	16.4k \pm 0.3k	17.9k \pm 0.2k (p=0.0727)
Inequality rule	IQ-RA1	18.6k \pm 0.4k (p=0.00119)	18.9k \pm 0.7k (p=0.0255)	19.1k \pm 0.1k (p=0.0341)	19.1k \pm 0.5k (p=0.0338)
	IQ-RA2	18.2k \pm 0.2k (p=0.0065)	18.1k \pm 0.5k (p=0.0021)	18.8k \pm 0.3k (p=0.0279)	19.0k \pm 0.1k (p=0.0332)
	IQ-RA3	19.3k \pm 0.4k (p=0.0138)	19.2k \pm 0.2k (p=0.0018)	19.5k \pm 0.3k (p=0.0275)	19.5k \pm 0.4k (p=0.0320)
	IQ-RA-E	18.5k \pm 0.1k (p=0.0129)	18.4k \pm 0.4k (p=0.0121)	18.8k \pm 0.3k (p=0.0116)	18.9k \pm 0.2k (p=0.0112)
Mixed rule	PL-RA2 + IQ-RA2	17.8k \pm 0.3k (p=0.0422)	17.5k \pm 0.4k (p=0.0113)	17.8k \pm 0.5k (p=0.0241)	18.8k \pm 0.1k (p=0.0348)
	PL-RA-E + IQ-RA-E	16.4k \pm 0.1k (p=0.0903)	16.2k \pm 0.2k (p=0.0667)	16.7k \pm 0.2k (p=0.0744)	17.8k \pm 0.2k (p=0.0919)

TABLE XI: FEs required to achieve $HV^T = 0.79$ for IEEE 300-bus system.

Rule Type	Repair agent	RU1	RU2	RU3	RU4
None	None (base)	20k (p=0.0141)	20k (p=0.0221)	20k (p=0.0316)	20k (p=0.0031)
Power law rule	PL-RA1	19.4k \pm 0.3k (p=0.0178)	19.1k \pm 0.1k (p=0.0110)	19.2k \pm 0.2k (p=0.0035)	19.8k \pm 0.1k (p=0.0063)
	PL-RA2	18.3k \pm 0.4k (p=0.0653)	18.0k \pm 0.1k (p=0.0318)	18.4k \pm 0.3k (p=0.0239)	18.9k \pm 0.4k (p=0.0157)
	PL-RA3	19.6k \pm 0.3k (p=0.0413)	19.3k \pm 0.2k (p=0.0212)	19.4k \pm 0.1k (p=0.0181)	19.4k \pm 0.3k (p=0.0025)
	PL-RA-E	17.5k \pm 0.5k	16.4k \pm 0.7k	17.7k \pm 0.6k	18.0k \pm 0.4k
Inequality rule	IQ-RA1	19.7k \pm 0.2k (p=0.0315)	19.5k \pm 0.1k (p=0.0033)	19.7k \pm 0.1k (p=0.0059)	19.8k \pm 0.1k (p=0.0024)
	IQ-RA2	19.6k \pm 0.1k (p=0.0122)	19.5k \pm 0.2k (p=0.0015)	19.5k \pm 0.1k (p=0.0073)	19.7k \pm 0.2k (p=0.0022)
	IQ-RA3	19.7k \pm 0.2k (p=0.0286)	19.8k \pm 0.1k (p=0.0074)	19.6k \pm 0.2k (p=0.0012)	19.8k \pm 0.1k (p=0.0016)
	IQ-RA-E	18.9k \pm 0.3k (p=0.0252)	18.2k \pm 0.2k (p=0.0104)	19.0k \pm 0.1k (p=0.0076)	19.2k \pm 0.2k (p=0.0032)
Mixed rule	PL-RA2 + IQ-RA2	18.0k \pm 0.4k (p=0.0991)	17.8k \pm 0.6k (p=0.0136)	18.0k \pm 0.4k (p=0.0503)	18.4k \pm 0.2k (p=0.0528)
	PL-RA-E + IQ-RA-E	17.7k \pm 0.3k (p=0.1013)	16.7k \pm 0.9k (p=0.0811)	17.8k \pm 0.3k (p=0.0713)	18.2k \pm 0.3k (p=0.661)



(a) ND Front for one run.



(b) HV plot over 20 runs.

Fig. 10: ND fronts and hypervolume plots obtained by IK-EMO with RU2 and power law repair agents for 1,100-member truss design problem.

TABLE XII: Variable groups for the 1,100-member truss cases. Each group has comparable variables having identical units and scales.

Group	Variable Type	Variable Indices
G_{t1}	l_i of vertical members	[1101 – 1179]
G_{t2}	r_i of top longitudinal members	[79 – 156], [235 – 312]
G_{t3}	r_i of bottom longitudinal members	[1 – 78], [157 – 234]
G_{t4}	r_i of vertical members	[313 – 391]

problem, ranking based on FEs to achieve the target HV for four users is summarized in Table XIV. An algorithm with statistically similar performance to the best performing algorithm is assigned a rank of 1. More details are presented in the supplementary document. It is seen that the top ranked

algorithm is PL-RA-E followed by the mixed PL-RA-E+IQ-RA-E, highlighting the superiority of the ensemble approach. PL-RA2 comes in the third place, showing that a moderate level of rule adherence provides the optimal performance.

VIII. COMPARISON WITH OTHER MOEAS

In the previous sections, the performance enhancement of NSGA-II with IK-EMO is investigated. A more complete analysis requires comparison with other learning based EMO algorithms which have attempted to learn and use the features of the search space or the fitness landscape properties [37]. Any such algorithm used for a performance comparison with IK-EMO-based NSGA-II should also be able to handle constrained and multi-objective optimization problems. Based on

TABLE XIII: FEs required to achieve $HV^T = 0.80$ for 1,100-member truss.

Rule Type	Repair agent	RU1	RU2	RU3	RU4
None	Base	1.0M (p=0.0083)	1.0M (p=0.0016)	1.0M (p=0.0101)	1.0M (p=0.0094)
Power law rule	PL-RA1	878.2k \pm 8.4k (p=0.0061)	856.4k \pm 8.6k (p=0.0131)	886.2k \pm 13.4k (p=0.0130)	889.2k \pm 6.0k (p=0.0025)
	PL-RA2	856.0k \pm 11.2k (p=0.0132)	730.2k \pm 15.4k (p=0.0270)	788.2k \pm 10.4k	803.2k \pm 17.8k
	PL-RA3	963.6k \pm 21.0k (p=0.0023)	>1.0M	>1.0M	>1.0M
	PL-RA-E	703.6k \pm 9.0k	684.4k \pm 18.8k	793.0k \pm 16.2k (p=0.1024)	810.0k \pm 24.0k (p=0.1151)
Inequality rule	IQ-RA1	963.0k \pm 15.2k (p=0.0060)	886.0k \pm 20.2k (p=0.0140)	872.0k \pm 23.0k (p=0.0154)	899.2k \pm 8.2k (p=0.0313)
	IQ-RA2	934.4k \pm 12.4k (p=0.0038)	842.4k \pm 19.6k (p=0.0243)	853.2k \pm 18.4k (p=0.0469)	902.0k \pm 6.0k (p=0.0339)
	IQ-RA3	>1.0M	979.0k \pm 29.0k (p=0.0212)	>1.0M	>1.0M
	IQ-RA-E	862.2k \pm 27.0k (p=0.0056)	853.2k \pm 22.2k (p=0.0319)	831.4k \pm 13.2k (p=0.0323)	854.4k \pm 11.8k (p=0.0215)
Mixed rule	PL-RA2 + IQ-RA2	712.4k \pm 10.8k (p=0.0689)	695.2k \pm 14.8k (p=0.0819)	802.0k \pm 12.4k (p=0.0776)	816k \pm 10.4k (p=0.0529)
	PL-RA-E + IQ-RA-E	705.2k \pm 9.4k (p=0.0722)	691.4k \pm 12.2k (p=0.0822)	797k \pm 13.0k (p=0.0613)	810.4k \pm 23.0k (p=0.0690)

TABLE XIV: Ranking of different repair agents on multiple problems. A detailed breakdown is provided in the supplementary material.

Problem	Base	PL-RA1	PL-RA2	PL-RA3	PL-RA-E	IQ-RA1	IQ-RA2	IQ-RA3	IQ-RA-E	PL-RA2 +IQ-RA2	PL-RA-E +IQ-RA-E
Beam 78-var	11	5	4	9	1	8	6	10	7	3	2
Beam 118-var	11	5	3	6	1	8	7	9	10	4	2
Power 115-var	11	4	3	6	1	9	7	10	8	5	2
Power 243-var	11	6	4	7	1	9	8	10	5	3	2
Truss 1179-var	9	7	2	10	1	8	6	10	5	4	2
Final Rank	11	5	3	8	1	9	6	10	7	4	2

these requirements, we have chosen three recently-proposed algorithms: RVEA [38], AGE-MOEA [39], and BiCo [40]. These algorithms use some form of adaptation methods to execute a more efficient search. PlatEMO [41], which provides a MATLAB implementation of these algorithms, is used here.

For IK-EMO, the parameters are kept the same as the previous experiments. PL-RA-E with RU2 is chosen as the repair scheme for IK-EMO, with NSGA-II as the core optimization algorithm. HV^T is set as the final median HV obtained by base NSGA-II.

Table XV shows the FEs required to achieve HV^T by all the algorithms for each problem. It is seen that IK-EMO with PL-RA-E statistically outperforms most of the other algorithms by reaching the HV^T faster. AGE-MOEA and BiCo are able to statistically match the IK-EMO's performance for only 78-variable beam and 115-variable OPF problems. The performance difference between IK-EMO and other methods is more significant for the large-scale 1,179-variable truss problem. It should be noted that the other algorithms have a better performance than the base NSGA-II on almost all problems. Thus, the superior performance of IK-EMO is not a product of the underlying NSGA-II alone, but of the effective VRG-based learning and repair methods as well. In addition, the user also has knowledge of functional relationships among design variables which can be used for different variants of the same problem. Thus, IK-EMO is a better choice over the other algorithms if knowledge interpretability and re-usability are desired and stays as a modular concept which can be embedded easily to other EMO/EMaO methods.

IX. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed the IK-EMO framework which interleaves interactive optimization with knowledge augmentation to obtain better quality solutions faster. Power

law, inequality, and mixed rules are extracted, together with their degrees of statistical adherence, from the ND solutions at a regular interval of generations. A computationally efficient graph data structure-based (VRG) knowledge processing method has been proposed to store and process multiple pairwise variable interactions. A user is then expected to provide a ranking of the learned rules based on his/her perception of the validity of the rules. A repair agent has been proposed to utilize the VRG with user-supplied ranking to repair offspring solutions. The study has created six repair schemes with three different degrees — tight, medium, and loose — of rule adherence. A mixed power law and inequality based repair has also been used. Finally, three ensemble-based repair schemes which adaptively use power law, inequality or both have been proposed. These 10 repair schemes have been implemented with four different rule usage schemes RU1-RU4, using 10% (conservative), to 100% (liberal) of the learned rules.

The proposed framework has been applied to three constrained large-scale two-objective practical problems: 78- and 118-variable stepped beam design problems, 115- and 243-variable optimal power flow problems, and a 1,179-variable truss design problem.

Experimental results on all problems have consistently shown that (i) usage of a moderate number of rules (20%) combined with a moderate degree of rule adherence produces a better performance compared to other individual repair operators, and (ii) power law rules, individually, produce better performance than inequality rules. Moreover, ensemble-based repair operators provide the best performance overall. Use of ensembles eliminates the need to experiment to find the right rule adherence for a new problem. IK-EMO is also able to work with very low population sizes, even for a large-scale problem. IK-EMO has also shown superior performance over other EMOs which adapt to the nature of the decision space or fitness landscape. IK-EMO can also help users learn about

TABLE XV: FEs required to achieve the HV^T for all the practical problems over 20 runs. HV^T is set as the final median HV of base NSGA-II. Problem-wise best-performing algorithm is marked in bold. Algorithms with statistically similar performance to the best algorithm for each problem are marked in italics. The corresponding Wilcoxon p-values are given in braces.

Problem	HV^T	NSGA-II	IK-EMO+PL-RA-E	RVEA	AGE-MOEA	BiCo
Beam 78-var	0.95	20.0k (p=0.0011)	14.6k \pm 0.1k	18.7k \pm 0.8k (p=0.0146)	<i>14.8k \pm 0.5k (p=0.0701)</i>	<i>14.8k \pm 0.3k (p=0.0614)</i>
Beam 118-var	0.76	20.0k (p=0.00121)	14.7k \pm 0.3k	19.3k \pm 0.4k (p=0.0210)	15.4k \pm 0.2k (p=0.0113)	15.2k \pm 0.4k (p=0.0405)
Power 115-var	0.90	20.0k (p=0.0013)	15.9k \pm 0.5k	19.6k \pm 0.8k (p=0.0098)	<i>16.3k \pm 0.3k (p=0.0563)</i>	<i>16.4k \pm 0.2k (p=0.0561)</i>
Power 243-var	0.79	20.0k (p=0.0056)	16.4k \pm 0.7k	>20k	18.1k \pm 0.5k (p=0.0297)	18.3k \pm 0.1k (p=0.0381)
Truss 1179-var	0.80	1.0M (p=0.0115)	684.4k \pm 18.8k	903.0k \pm 12k (p=0.0013)	723.0k \pm 22k (p=0.0021)	717.4 \pm 12k (p=0.0488)

unknown relations between the decision variables.

This study opens up a number of avenues for future work. The scope of knowledge can be increased to include the objective and constraint functions as well, in line with existing innovization literature. Studies can be performed using rule types other than the ones considered here. For EMO algorithms using some form of adaptation mechanisms, repair methods can interfere with the natural evolution of optimal solutions. Studies need to be performed to understand how the repair agents affect the operation of adaptation-based algorithms. In this work, the learning and repair intervals are kept fixed. The effect of these parameters need to be studied more closely. In many problems, a rule may not stay valid across the entire Pareto-optimal front. Locally present rules may exist in certain parts of the Pareto-optimal front [42]. Ways to extract local rules and repair a MOEA's offspring population members accordingly will introduce additional challenges but may result in faster convergence. The interactive and knowledge-based approach should also be implemented to other EMO/EMaO algorithms to observe its effect in improving convergence speed.

Traditional user preference information including, but not limited to, relative importance of objective functions and preferred regions of the Pareto-optimal front, can also potentially be integrated into this type of framework. Nevertheless, this study has clearly demonstrated a viable way to extract variable interaction knowledge from intermediate optimization iterations and to use relevant and vetted knowledge back in the optimization algorithm for updating offspring solutions to constitute a computationally fast search process. More such practice-oriented studies must now accompany evolutionary optimization applications to make them more worthy for practical problem solving tasks.

REFERENCES

- [1] K. Deb and R. Bhushan Agrawal, "Simulated Binary Crossover for Continuous Search Space," *Complex Syst.*, vol. 9, no. 2, pp. 115–148, 1995.
- [2] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, 2005.
- [3] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, mar 2000.
- [4] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, vol. 1. IEEE Computer Society, 2002, pp. 825–830.
- [5] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," in *Lecture Notes in Computer Science*, vol. 3410. Springer Verlag, 2005, pp. 280–295.
- [6] K. Deb and C. Myburgh, "Breaking the Billion-Variable Barrier in real-world optimization using a customized evolutionary algorithm," in *GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, New York, NY, USA, jul 2016, pp. 653–660.
- [7] A. Szollos, M. Šmíd, and J. Hájek, "Aerodynamic optimization via multi-objective micro-genetic algorithm with range adaptation, knowledge-based reinitialization, crowding and ϵ -dominance," *Advances in Engineering Software*, vol. 40, no. 6, pp. 419–430, jun 2009.
- [8] A. H. Gandomi, K. Deb, R. C. Averill, S. Rahnamayan, and M. N. Omidvar, "Using semi-independent variables to enhance optimization search," *Expert Systems with Applications*, vol. 120, pp. 279–297, 2019.
- [9] R. Landa-Becerra, L. V. Santana-Quintero, and C. A. Coello, "Knowledge incorporation in multi-objective evolutionary algorithms," *Studies in Computational Intelligence*, vol. 98, pp. 23–46, 2008.
- [10] C. Barba-González, A. J. Nebro, J. García-Nieto, M. del Mar Roldán-García, I. Navas-Delgado, and J. F. Aldana-Montes, "Injecting domain knowledge in multi-objective optimization problems: A semantic approach," *Computer Standards & Interfaces*, vol. 78, p. 103546, 10 2021.
- [11] C. A. C. Coello and M. G. C. Tapia, "Cultural Algorithms for Optimization," in *Handbook of AI-based Metaheuristics*. CRC Press, jul 2021, pp. 219–238.
- [12] S. Obayashi and D. Sasaki, "Visualization and data mining of Pareto solutions using Self-Organizing Map," *Lecture Notes in Computer Science*, vol. 2632, pp. 796–809, 2003.
- [13] K. Deb and A. Srinivasan, "Innovization: Innovating design principles through optimization," in *GECCO 2006 - Genetic and Evolutionary Computation Conference*, vol. 2, 2006, pp. 1629–1636.
- [14] S. Bandaru and K. Deb, "Higher and lower-level knowledge discovery from Pareto-optimal sets," in *Journal of Global Optimization*, vol. 57, no. 2. Springer, oct 2013, pp. 281–298.
- [15] A. Gaur and K. Deb, "Adaptive use of innovization principles for a faster convergence of evolutionary multi-objective optimization algorithms," in *Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, New York, USA, jul 2016, pp. 75–76.
- [16] B. Xin, L. Chen, J. Chen, H. Ishibuchi, K. Hirota, and B. Liu, "Interactive multiobjective optimization: A review of the state-of-the-art," pp. 41 256–41 279, jul 2018.
- [17] K. Deb and J. Sundar, "Reference point based multi-objective optimization using evolutionary algorithms," in *GECCO 2006 - Genetic and Evolutionary Computation Conference*, vol. 1. New York, New York, USA: ACM Press, 2006, pp. 635–642.
- [18] K. Miettinen, F. Ruiz, and A. P. Wierzbicki, "Introduction to multiobjective optimization: Interactive approaches," in *Lecture Notes in Computer Science*. Springer Verlag, 2008, vol. 5252 LNCS, pp. 27–57.
- [19] M. Gombolay, R. Jensen, J. Stigile, T. Golen, N. Shah, S. H. Son, and J. Shah, "Human-machine collaborative optimization via apprenticeship scheduling," *Journal of Artificial Intelligence Research*, vol. 63, pp. 1–49, may 2018.
- [20] S. Bandaru, T. Aslam, A. H. Ng, and K. Deb, "Generalized higher-level automated innovization with application to inventory management," *European Journal of Operational Research*, vol. 243, no. 2, pp. 480–496, jun 2015.
- [21] K. Deb and R. Datta, "Hybrid evolutionary multi-objective optimization and analysis of machining operations," *Engineering Optimization*, vol. 44, no. 6, pp. 685–706, jun 2012.
- [22] A. H. Ng, C. Dudas, H. Boström, and K. Deb, "Interleaving innovization with evolutionary multi-objective optimization in production system simulation for faster convergence," in *Lecture Notes in Computer Science*, vol. 7997 LNCS. Springer, Berlin, Heidelberg, 2013, pp. 1–18.
- [23] A. Ghosh, K. Deb, R. Averill, and E. Goodman, "Combining User Knowledge and Online Innovization for Faster Solution to Multi-objective Design Optimization Problems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and*

- Lecture Notes in Bioinformatics*), vol. 12654 LNCS. Springer, Cham, mar 2021, pp. 102–114.
- [24] A. Ghosh, K. Deb, E. Goodman, and R. Averill, “A user-guided innovization-based evolutionary algorithm framework for practical multi-objective optimization problems,” *Engineering Optimization*, vol. 0, no. 0, pp. 1–13, 2022. [Online]. Available: <https://doi.org/10.1080/0305215X.2022.2144275>
 - [25] A. Ghosh, E. Goodman, K. Deb, R. Averill, and A. Diaz, “A Large-scale Bi-objective Optimization of Solid Rocket Motors Using Innovization,” *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, jul 2020.
 - [26] A. Gaur and K. Deb, “Adaptive use of innovization principles for a faster convergence of evolutionary multi-objective optimization algorithms,” in *Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, New York, New York, USA, jul 2016, pp. 75–76.
 - [27] C. Barba-González, V. Ojalehto, J. García-Nieto, A. J. Nebro, K. Miettinen, and J. F. Aldana-Montes, “Artificial decision maker driven by PSO: An approach for testing reference point based interactive methods,” in *Lecture Notes in Computer Science*, vol. 11101 LNCS. Springer Verlag, 2018, pp. 274–285.
 - [28] A. V. Aho, M. R. Garey, and J. D. Ullman, “The Transitive Reduction of a Directed Graph,” *SIAM Journal on Computing*, vol. 1, no. 2, pp. 131–137, jul 1972.
 - [29] A. Rothwell, “Optimization of beams,” in *Solid Mechanics and its Applications*. Springer Verlag, apr 2017, vol. 242, pp. 147–181.
 - [30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, apr 2002.
 - [31] E. Zitzler and L. Thiele, “Multiobjective optimization using evolutionary algorithms - A comparative case study,” in *Lecture Notes in Computer Science*, vol. 1498 LNCS. Springer Verlag, 1998, pp. 292–301.
 - [32] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric Statistical Methods*, ser. Wiley Series in Probability and Statistics. Wiley, jul 2015.
 - [33] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., 2001.
 - [34] S. Datta, A. Ghosh, K. Sanyal, and S. Das, “A Radial Boundary Intersection aided interior point method for multi-objective optimization,” *Information Sciences*, vol. 377, pp. 1–16, 2017.
 - [35] M. Basu, “Economic environmental dispatch using multi-objective differential evolution,” *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 2845–2853, 2011.
 - [36] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, “MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education,” *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, feb 2011.
 - [37] Y. E. Tian, L. Si, X. Zhang, K. Chen, Y. Jin, . R. Cheng, C. He, G. Provincial, . K. C. Tan, . Y. Jin, Y. Tian, R. Cheng, C. He, and K. C. Tan, “Evolutionary large-scale multi-objective optimization: A survey,” *ACM Computing Surveys (CSUR)*, vol. 54, 10 2021.
 - [38] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, “A reference vector guided evolutionary algorithm for many-objective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 20, pp. 773–791, 10 2016.
 - [39] A. Panichella, “An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization,” *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference*, pp. 595–603, 7 2019. [Online]. Available: <https://doi.org/10.1145/3321707.3321839>
 - [40] Z. Z. Liu, B. C. Wang, and K. Tang, “Handling constrained multi-objective optimization problems via bidirectional coevolution,” *IEEE Transactions on Cybernetics*, 10 2021.
 - [41] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, “Platemo: A matlab platform for evolutionary multi-objective optimization,” 2017. [Online]. Available: <https://arxiv.org/abs/1701.00879>
 - [42] S. Bandaru and K. Deb, “Higher and lower-level knowledge discovery from pareto-optimal sets,” *Journal of Global Optimization*, vol. 57, no. 2, pp. 281–298, 2013.