An Innovization-based Evolutionary Algorithm Framework for Large-Scale Practical Multi-Objective Optimization Problems

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

COIN Report Number 2021004

***Abstract*—Large-scale multi-objective optimization problems (LSMOPs) are challenging to solve for most optimization algorithms including standard and generic multi-objective evolutionary algorithms (MOEAs) due to the "curse of dimensionality." Although there exist a few case studies involving *customized* evolutionary algorithms to solve large-scale problems, specific problem knowledge obtained through extensive experience of users or domain experts can facilitate a more efficient application. *Innovization* is the process of uncovering patterns that exist among good solutions. Any common pattern extracted from good solutions discovered during an optimization run can be used to modify candidate solutions in the form of a repair operator, but the key aspect is to strike a balance between the relevance of the pattern identified and the extent of their use in the repair operator, lest the learned patterns are properties of artificially-stuck solutions. Thus, the use of problem knowledge for the initial guidance and innovization-based repair of solutions during optimization in an online fashion can help find better solutions, faster. This paper proposes a customized MOEA framework which combines problem-specific knowledge and online innovization methods to solve two real-world LSMOPs – 879 and 1,479-variable truss structure design and 544-variable solid fuel rocket design. Four different repair operators are proposed that are suitable for uncovering monotonic relations involving multiple decision variables. The performance variations resulting from different combinations of initial user knowledge and repair operators have also been studied. The proposed approach is generic and provides a viable direction for handling large-scale multi-objective practical problems.**

***Index Terms*—Large-scale problems, multi-objective optimization, 'innovization', knowledge-based optimization.**

I. Introduction

LARGE-SCALE multi-objective optimization problems (LSMOPs) can pose a challenge to standard classical or multi-objective evolutionary algorithms (MOEAs) [1]. Increasing numbers of decision variables can result in deteriorating performance for MOEAs [2], [3], an issue caused by the well-known 'curse of dimensionality' first coined by Bellman [4] in the context of single-variable dynamic optimization problems. Furthermore, many practical optimization problems can be considered large-scale due to the large number of decision variables that they possess. Besides large-scale single-objective problems [5], [6], some large-scale multi-objective problems addressed by researchers are portable water distribution network design [7], neural architecture search [8], ratio error estimation of voltage transformers [9], truss-structure design [10], and complex network reconstruction [11], [12].

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).

Many researchers have attempted to mitigate the shortcomings of MOEAs on large-scale problems using different strategies. Cooperative co-evolution, (CC) [13] a popular approach, decomposes a large-scale problem into a number of smaller sub-problems which are relatively easy to solve. Many researchers have adapted CC-based approaches for use in MOEAs, such as LMEA [14], tested on upto 5,000 variables, DPCCMOLSIA [15] (up to 1,000 variables), [16] (up to 200 variables), [17] (5,000 variables), [18] (1,200 variables), and [19] (5,000 variables). A scalable indicator-based MOEA was proposed in [20] (up to 1,000 variables). An algorithm for solving large-scale multi-modal MOEAs was proposed in [8] (100 variables).

In terms of large-scale problem solving, many MOEAs used benchmark problem suites, such as ZDT [21], DTLZ [22], WFG [23], and CEC 2009 [24] problems, which are scalable in terms of decision variables. Such studies are good for assessing the scalable performance of MOEAs, but they have been criticized for the algorithm's ability to exploit the structured patterns present in the optimal solutions for such benchmark problems. Thus, the real test of the scalability of MOEAs should come from directly solving large-scale practical problems.

It has been discussed in the context of single-objective EAs that an optimization algorithm with generic recombination and mutation operators (such as simulated binary crossover (SBX) [25] or differential evolution (DE) [26]) may be too slow to lead to high-performing regions of the search space. Although they are probably ideal for solving benchmark problems, due to no other problem specific information being available for them, real-world problems usually come with physical parameters that must be related in certain ways for a solution to be meaningful. The use of additional problem information, if available, can be used to modify the operators so that they are customized for the problem. In a recent billion-variable resource allocation problem originated from a casting scheduling task in a foundry [5], the linearity of constraint structures motivated the authors to develop a recombination and two mutation operators that exploit the linearity. The outcome has demonstrated a polynomial performance of the resulting customized genetic algorithm that scales to solve similar scheduling problems having 50,000 to one billion variables.

Another study used the concept of semi-independent variables [27], in which a redefinition of variables was proposed to handle user-specified monotonic relationships among variables in the form of $x_i \leq x_{i+1} \leq x_{i+2} \leq \ldots \leq x_j$ as additional knowledge, so that every created solution automatically satisfied all the above constraints. However, for a different relationship (such as a monotonically decreasing relationship, or another pattern in variables), a new description of semi-independent variables needs to be defined and implemented.

Importantly, users who are interested in solving large-scale and complex real-world problems have many years of expertise and knowledge in formulating some useful relationships among variables for a generic high-performing solution for the problem. Often, academic research tends to ignore such information in devising an algorithm, anticipating that the use of such additional information may 'dilute' the overall study. However, when real-world problems are to be solved in practice, it is perfectly legitimate to use any such additional information that is available to help solve the problem class routinely. In fact, not using vital information that is available to assist in the problem solving task may be considered 'insulting' to the user's many years of earned knowledge and intuition about the problem.

Whether the study is academic or real-world, it can be beneficial to use any available problem information (in addition to the objectives, constraints, variable bounds, etc. that define the optimization problem) within the optimization algorithm. This may allow a faster and more efficient search for an optimized solution. In this case, several challenging questions remain, for example: How do we incorporate additional problem information into an algorithm? How frequently and to what extent should we influence the algorithm with the known problem information? These questions arise because the additional information may strongly influence the properties of the final solutions. If this information is somehow incomplete or imperfect, then allowing too much influence right from the beginning of an optimization run may steer the overall process away from the optimal solutions, or may lead to a premature convergence to sub-optimal solutions. Thus, the use of experience or intuition-based additional problem information provided at the start of the optimization task need not be considered as 'given' constraints or properties, but rather an adaptive algorithm must assess their worth systematically and utilize them differently as the algorithm navigates from its initial generation to the end. Besides, recent *innovization* studies [28], [29] have demonstrated that additional problem information can also be extracted from high-performing solutions created during the optimization process. The so-called 'innovized' properties, which are developed by the algorithm, can be ideally compared and contrasted with supplied additional problem information to determine the worth of such problem information at various stages in an optimization run. In this paper, we address some of these issues and propose generic knowledge-based methodologies for solving large-scale multi-objective optimization problems.

The remainder of the paper is organized as follows. Section II covers the *innovization* concept which makes up the core of this paper. Section III presents the proposed framework along with the different innovization-based repair operators. Sections IV, V and VI present the formulation and experimental results for the two practical problems considered in this paper. Section VII concludes the paper and suggests a number of future studies.

II. Innovization Task

MOEAs are applied to a wide variety of design problems, producing a set of Pareto-optimal (PO) solutions that represent a trade-off between the relevant objectives. It is then left to the decision-maker to choose a particular solution. Within these PO solutions, there may exist implicit relationships or common characteristics among design variables. *Innovization* is the process of extracting such patterns, first proposed by Deb and Srinivasan [28]. There, the authors presented a method to cluster together diverse solutions obtained at the end of an NSGA-II run, and analyze them for commonalities. The approach was tested on multiple practical problems such as multiple-disk clutch brake design, spring design, and welded beam design.

Innovization is a general process and multiple implementations exist in the literature. [29] introduced the concept of higher and lower level innovization. The former involves learning common characteristics present among solutions from multiple Pareto fronts, whereas, lower level innovization involves learning from a specific portion of a single Pareto front. A niched clustering-based optimization technique was proposed in [30] to automate the innovization process. A genetic programming-based innovization framework was proposed in [31] and was applied on an inventory management problem.

Recent studies have proposed methods to perform innovization in an online manner, and inject the findings back into the optimization process as a heuristic. An MOEA combined with a local search procedure was employed in [32] to optimize different machining parameters and ensure faster convergence through online innovization. A combination of innovization and data mining approaches were used in [33] to achieve faster convergence. Gaur and Deb [34] 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 variant of the approach for problems with discrete variables also exists [35].

The basic principle of innovization is to generate rules in simple forms such as power laws ($x_i x_j^b = c$). In [31], the authors have proposed a method which is able to express relationships involving operators like summation $(+)$, difference $(-)$, product $(\times)$, etc. In this paper, we primarily focus on relations consisting of equalities $(=)$, usually employed to express some form of symmetric relations, or inequalities $(\leq, \geq)$, which can be used to uncover patterns where certain groups of variables monotonically increase or decrease. However, the framework presented in this paper can also be adapted for any type of relations provided a suitable repair operator can also be designed to exploit them.

III. Multi-objective Optimization Algorithm with Innovization-based Repair (IR)

A. *User knowledge specification*

For an optimization problem with n decision variables, there can exist a large number of correlations among two or more variables for high-performing (or near-optimal) solutions. For simplicity, we will stay with the minimal possible variable interactions and limit our scope to pairwise relations. However, a chain of such relationships will structurally extend to exhibit relationships among more than two variables indirectly. To systematically express the pre-specified user knowledge, we make use of an upper diagonal $n \times n$ relationship matrix $\mathbf{U}$, as shown below:

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix}, \tag{1}$$

in which u_{ij} $(i \leq j)$ indicates the type of relationship between variables x_i and x_j. We consider five different relationships, as below:

$$u_{ij} = \begin{cases} 0, & \text{if } x_i \text{ and } x_j \text{ are not likely to have a relationship,} \\ 1, & \text{if } x_i \text{ and } x_j \text{ have a relationship, but unknown,} \\ 2, & \text{if } x_i \leq x_j, \\ 3, & \text{if } x_i \geq x_j, \\ 4, & \text{if } x_i \approx x_j. \end{cases} \tag{2}$$

Clearly, $u_{ii} = 4$. For LSMOPs, a large number of decision variables will make specifying all $\binom{n}{2}$ relationships extremely cumbersome. However, in large-scale practical problems, variable patterns can be specified or envisioned to have relationships in groups. K groups of variables can be specified (G_k, $k = 1, 2, \ldots, K$) with common variables among groups. The following $\mathbf{U}$ contains three groups of variables $G_1 = \{1, 2, 3\}$, $G_2 = \{1, 5, 6\}$ and $G_3 = \{4, 6\}$:

$$\mathbf{U} = \begin{bmatrix} 4 & 3 & 3 & 0 & 2 & 2 \\ & 4 & 3 & 0 & 0 & 0 \\ & & 4 & 0 & 0 & 0 \\ & & & 4 & 0 & 1 \\ & & & & 4 & 2 \\ & & & & & 4 \end{bmatrix}$$

indicating following relationships: $x_1 \geq x_2 \geq x_3$, $x_1 \leq x_5 \leq x_6$ and (x_4, x_5) are related but with unknown relationship. There is redundant information in the matrix. For example, $u_{13} = 3$ is already implied from $u_{12} = 3$ and $u_{23} = 3$. Thus, a consistency check of the matrix is important, which can be performed using simple matrix operations.

In order to estimate the extent of knowledge specified by the user, a metric ($\mathcal{K}$) is proposed here which calculates the number of relations specified as a proportion of the total number of possible relations. Here, 'crisp' or explicit knowledge ($u_{ij} = 4$) is given a weight of one, partial information with $u_{ij} = 2$ or 3 is assigned a weight of 0.75, and 'inexact' knowledge ($u_{ij} = 1$) is given a weight of 0.5. Albeit somewhat arbitrary, the following $\mathcal{K}$ measure is used as an indication of overall problem information provided to the algorithm:

$$\mathcal{K} = \frac{\sum\limits_{i<j}[u_{ij} = 4] + 0.75\sum\limits_{i<j}[u_{ij} = 2 \vee 3] + 0.5\sum\limits_{i<j}[u_{ij} = 1]}{n(n-1)/2}. \tag{3}$$

B. *Proposed MOEA/I framework*

The proposed MOEA with innovization-based repair (MOEA/I) framework is shown in Fig. 1. It shows the entirety of the optimization process, including the problem formulation phase. The major components are:

- Problem specification - This is the first step in the optimization. Here, the user specifies the objective functions, the decision variables, the constraints, and the model to be used. The user and the optimization algorithm designer can also work together to collect problem-specific information and express them in a format easily usable by the algorithm.
- Optimization strategy design - Here, the optimization algorithm designer chooses or creates a suitable algorithm to be used for the problem. In addition, if innovization is to be performed, the corresponding procedure needs to be specified at this step.
- Innovization-based MOEA (MOEA/I) - This is the customized optimization algorithm to be used for this problem. It has the following major components:
  - Generating new solutions - New solutions are generated from the parent population using reproduction operators such as crossover and mutation.
  - Innovization block - Perform an innovization study on good solutions in the parent population and extract patterns in the form of 'innovization rules'. The rules can be expressed as another matrix $\mathbf{L}$ whose elements are assigned in the same fashion as the relationship matrix $\mathbf{U}$, but is based on the best solutions found so far.
  - Repair block - Modify new solutions according to the innovization rules learned previously.
  - Evaluation - Evaluate the objective functions for the repaired solution set.

The MOEA/I optimization procedure shown in Fig. 1 is similar to commonly-used MOEAs like NSGA-II. The only modification is the innovization process implemented through the innovization and repair blocks. The former is used to learn innovization rules and the latter applies them to the offspring population.

The innovization block operation is shown in Algorithm 1. We cross-check every pairwise variable set in a group G_k with the relationship matrix $\mathbf{U}$ to determine which variable pairs are designated by the user to follow specific relations ($u_{ij} =$ 2, 3 or 4) and which ones need to be identified ($u_{ij} = 1$) for their relationships through the innovization procedure. For these variables, a reference vector ($\mathbf{x}_{ref}$) is calculated from the best solutions obtained so far. The procedure used for this is covered in the subsequent sections. A score is assigned equal to the proportion of the population that follow the relationship

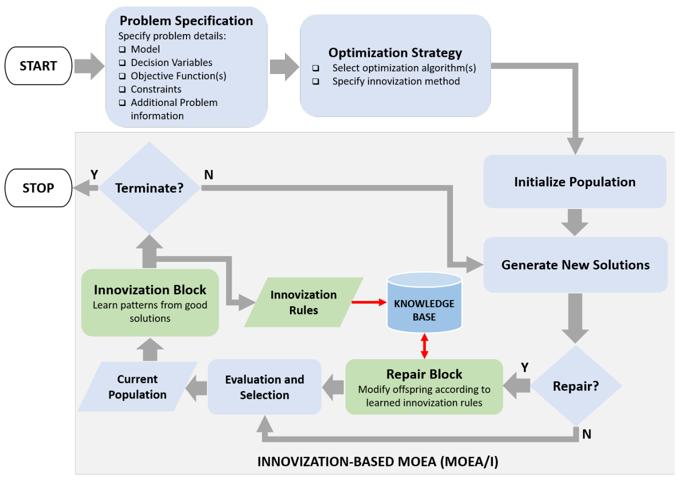


Fig. 1: Innovization-based MOEA Framework (MOEA/I)

Algorithm 1 Innovization block

```
Require: Best population $\mathcal{X}$, population size (N), reference vector ($\mathbf{x}_{ref}$), variable groups ($G$), user relation matrix ($\mathbf{U}$), innovization rules ($\mathbf{L}$)
Ensure: Updated reference vector ($\mathbf{x}_{ref}$), innovization rules ($\mathbf{L}$), rule scores ($\mathbf{P}$)
1:  Calculate reference vector ($\mathbf{x}_{ref}$) from good solutions;
2:  for each group $G_k$ do
3:      for each variable pair $(i, j)$ in $G_k$ do
4:          $p_l \leftarrow 0$;                    ▷ Rule score for $x_i \leq x_j$
5:          $p_g \leftarrow 0$;                    ▷ Rule score for $x_i \geq x_j$
6:          $p_e \leftarrow 0$;                    ▷ Rule score for $x_i \approx x_j$
7:          $p_{max} \leftarrow 0$;                ▷ Store the highest rule score
8:          if $\mathbf{U}_{ij} = 1$ then
9:              for each solution $\mathbf{x}$ in $\mathcal{X}$ do
10:                 if $x_i \leq x_j$ and $x_{i_{ref}} \leq x_{j_{ref}}$ then
11:                     $p_l \leftarrow p_l + \frac{1}{N}$;
12:                 else if $x_i \geq x_j$ and $x_{i_{ref}} \geq x_{j_{ref}}$ then
13:                     $p_g \leftarrow p_g + \frac{1}{N}$;
14:                 end if
15:                 if $x_i \approx x_j$ and $x_{i_{ref}} \approx x_{j_{ref}}$ then
16:                     $p_e \leftarrow p_e + \frac{1}{N}$;
17:                 end if
18:             end for
19:             $p_{max} \leftarrow \max(p_l, p_g, p_e)$;
20:             if $p_{max} = p_l$ then
21:                 $\mathbf{L}_{ij} \leftarrow 2$;
22:             else if $p_{max} = p_g$ then
23:                 $\mathbf{L}_{ij} \leftarrow 3$;
24:             else if $p_{max} = p_e$ then
25:                 $\mathbf{L}_{ij} \leftarrow 4$;
26:             end if
27:         else
28:             $\mathbf{L}_{ij} \leftarrow \mathbf{U}_{ij}$;
29:             $p_{max} \leftarrow 1$;
30:         end if
31:         $\mathbf{P}_{ij} \leftarrow p_{max}$;
32:     end for
33: end for
```

existing among the concerned variable pairs. This score acts as the probability with which a particular offspring is repaired.

The repair block operation is shown in Algorithm 2. Here, for every variable group, if an explicit relation is defined, the repair operator ensures every offspring follows them. Otherwise, the relationships learned through innovization are applied with the probability calculated in the innovization block.

The subsequent sections present the four repair operators used in this paper, three of which are an extension of the operators proposed in [36]. Each repair operator functions differently when explicit relations are specified compared to when only variable groups are specified without any explicit relation. In this paper, the scope of the repair operators are kept limited to inequality relations. However, if the problem demands it, custom repair operators can be designed and plugged into the MOEA/I framework. Every repair operator uses the knowledge available during the optimization run to different extents. For example, operator IR1 constrains the offsprings the least, whereas operator IR3 does the most. In addition, an ensemble operator (I-ES) is also proposed which automatically switches between IR1, IR2 and IR3 based on their individual performances. Three repair operators from IR1 to IR3 are described in detail in Table I.

C. MOEA/IR1 Procedure

It is assumed that the user has already provided the problem information matrix $\mathbf{U}$ with different groups of variables and their relationship type. MOEA/IR1 computes the variable-wise average x_i (or $x_{i_{avg}}$) from all non-dominated (ND) solutions at a generation and repairs a pair of an offspring solutions (x_i and x_j) based on the supplied problem information u_{ij}. MOEA/IR1 uses the information of the current population and does not collaborate with the same of the past generation. In this sense, it uses instantaneous information and may not be trustworthy.

The repaired variable value x_{ir} for $u_{ij} = 1$, 2, and 3 are shown in Table I. If $u_{ij} = 0$, no repair is performed and a free-form evolution is allowed. If $u_{ij} = 1$ meaning that a relation is expected, but is unknown, MOEA/IR1 attempts to learn the evolved relationship between x_i and x_j present in the non-dominated (ND) solutions and enforce to repair both variable values. If $x_{i_{avg}}$ is smaller than $x_{j_{avg}}$ and x_i is also smaller than x_j, the current (x_i,x_j) pair matches the relationship among ND solutions and hence, (x_i,x_j) pair is not modified. However, if $x_i \geq x_j$, disagreeing with the found relationship between their average ND values, the repaired (x_{ir}, x_{j_r}) pair in Table I are closer to their ($x_{i_{avg}}$,$x_{j_{avg}}$) values. The difference $|x_{i_{avg}} - x_{ir}| = |x_{i_{avg}} - 0.5(x_{i_{avg}} + x_i)| = 0.5|x_{i_{avg}} - x_i|$, which is smaller than the original difference $|x_{i_{avg}} - x_i|$ by 50%. The same is true for repaired variable x_{j_r}.

For $u_{ij} = 2$ and 3, the x_i and x_j are repaired carefully (shown in Table I) so that the supplied relationship among the two variable is attempted to achieve. For $u_{ij} = 4$, $x_{i_r} = x_{j_r} = \text{random}(x_{i_{avg}}, x_{j_{avg}})$ is assigned.

TABLE I: Repair operator description.

Operator	Unspecified relation $(u_{ij} = 1)$	Explicitly specified relation $(u_{ij} = [2,3])$
IR1	$x_{i_r} = \begin{cases} \frac{x_{i_{avg}}+x_i}{2}, & \text{for } (x_{i_{avg}} - x_{j_{avg}})(x_i - x_j) < 0, \\ x_i, & \text{otherwise.} \end{cases}$ $x_{j_r} = \begin{cases} \frac{x_j+x_{j_{avg}}}{2}, & \text{for } (x_{i_{avg}} - x_{j_{avg}})(x_i - x_j) < 0, \\ x_j, & \text{otherwise.} \end{cases}$	$x_{i_r} = \begin{cases} x_{ij_{avg}} - \frac{\lvert x_{ij_{avg}} - x_{i_{avg}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{avg}} + \frac{\lvert x_{ij_{avg}} - x_{i_{avg}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ $x_{j_r} = \begin{cases} x_{ij_{avg}} + \frac{\lvert x_{ij_{avg}} - x_{j_{avg}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{avg}} - \frac{\lvert x_{ij_{avg}} - x_{j_{avg}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ where $x_{ij_{avg}} = \frac{x_{i_{avg}} + x_{j_{avg}}}{2}$.
IR2	$x_{i_r} = \begin{cases} \frac{x_{i_{ref}}+x_i}{2}, & \text{for } (x_{i_{ref}} - x_{j_{ref}})(x_i - x_j) < 0, \\ x_i, & \text{otherwise.} \end{cases}$ $x_{j_r} = \begin{cases} \frac{x_j+x_{j_{ref}}}{2}, & \text{for } (x_{i_{ref}} - x_{j_{ref}})(x_i - x_j) < 0, \\ x_j, & \text{otherwise.} \end{cases}$ where $x_{i_{ref}}(t) = x_{i_{avg}}(t) + \gamma \left(x_{i_{avg}}(t) - x_{i_{avg}}(t-1) \right)$	$x_{i_r} = \begin{cases} x_{ij_{ref}} - \frac{\lvert x_{ij_{ref}} - x_{i_{ref}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{ref}} + \frac{\lvert x_{ij_{ref}} - x_{i_{ref}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ $x_{j_r} = \begin{cases} x_{ij_{ref}} + \frac{\lvert x_{ij_{ref}} - x_{j_{ref}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{ref}} - \frac{\lvert x_{ij_{ref}} - x_{j_{ref}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ where $x_{i_{ref}}(t) = x_{i_{avg}}(t) + \gamma \left(x_{i_{avg}}(t) - x_{i_{avg}}(t-1) \right)$
IR3	$x_{i_r} = \mathcal{U}(x_{i_{ref}} - \sigma_i, x_{i_{ref}} + \sigma_i),$ $x_{j_r} = \mathcal{U}(x_{j_{ref}} - \sigma_j, x_{j_{ref}} + \sigma_j),$ where $\mathcal{U}(a,b) \equiv$ Uniform distribution between [a,b]	$x_{i_r} = \begin{cases} x_{ij_{ref}} - \frac{\lvert x_{ij_{ref}} - x_{i_{ref}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{ref}} + \frac{\lvert x_{ij_{ref}} - x_{i_{ref}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ $x_{j_r} = \begin{cases} x_{ij_{ref}} + \frac{\lvert x_{ij_{ref}} - x_{j_{ref}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{ref}} - \frac{\lvert x_{ij_{ref}} - x_{j_{ref}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ where $x_{i_{ref}}(t) = x_{i_{avg}}(t) + \gamma \left(x_{i_{avg}}(t) - x_{i_{avg}}(t-1) \right)$

Algorithm 2 Repair block

```
Require: New solution set $\mathcal{X}$, population size (N), user knowledge matrix ($\mathbf{U}$), variable groups ($G$), repair operator ($R$), innovization rules ($\mathbf{L}$), repair probabilities ($\mathbf{P}$)
Ensure: Repaired solution set $\mathcal{X}_r$
1:  for each group $G_k$ do
2:      for each variable pair $(i, j)$ in $G_k$ do
3:          for each solution $\mathbf{x}$ in $\mathcal{X}$ do
4:              Generate a random number $r$ between 0 and 1;
5:              if $r \leq \mathbf{P}_{ij}$ then
6:                  if $\mathbf{U}_{ij} = 1$ then
7:                      Use repair operator R according to learned relation $\mathbf{L}_{ij}$;
8:                  else
9:                      Use repair operator R according to explicit relation $\mathbf{U}_{ij}$;
10:                 end if
11:             end if
12:         end for
13:     end for
14: end for
```

D. MOEA/IR2 Procedure

This repair operator follows a similar repair process as that of MOEA/IR1, except that $x_{i_{avg}}$ is replaced with $x_{i_{ref}}$, which uses a history of change of the average x_i from past to the current generation, as shown in Table I. The notable difference between the IR1 and IR2 operators is the inclusion of a momentum parameter (γ), noting that $\gamma = 0$ makes both methods identical. The parameter is intended to provide a boost to the optimization algorithm towards a projected good solution. The average value $\mathbf{x}_{avg}$ of the variables under consideration reflect the approximate pattern followed by good solutions in the current generation. The momentum term takes into account the average values in the previous generation $\mathbf{x}_{i_{avg}}(t-1)$, and allows for faster convergence towards optimal solutions. It also allows the algorithm to avoid following $\mathbf{x}_{avg}$ too closely, thus, preserving diversity. This repair method is more trustworthy than IR1, as an attempt is made to make the variable values close to historically agreeable average value, rather than current average value alone. The update of variables for $u_{ij} = 4$ is identical to that in IR1.

E. MOEA/IR3 Procedure

This repair operator is similar to IR2 for $u_{ij} = 2$ and 3, but for $u_{ij} = 1$, the values are repaired to be within one-sigma (σ_i is the standard deviation of x_i values among ND solutions of current generation) away from the historical average point. Thus, this method trusts the observed relationships of x_i and x_j more closely than IR1 and IR2. The update of variables for $u_{ij} = 4$ is identical to that in IR1 and IR2.

F. Ensembled NSGA-II/I-ES Procedure

All repair operators designed for MOEA/I needs to be tested through separate experiments. However, in practical applications, such experiments might prove costly. A method to avoid this is to combine these operators into an ensemble. During the optimization run, the performance of each repair operator is tracked, and the number of solutions allowed to be repaired by each operator is based on the historical performance of the offsprings generated by each. This relieves the need of analyzing every operator since the best will be selected automatically. The ensemble method also considers the base NSGA-II as a repair operator, which represents the case when the offspring is not repaired. This allows the optimization to remain unaffected if bad repair operators are supplied. Thus, a total of four repair operators are used in the ensemble process.

The performance of each offspring generated by the i-th repair operator is based on its offspring survival rate (r_s^i). Greater the survival rate of an operator, greater is the probability of repair. The probability (p_r^i) update operation for i-th operator is described below:

$$p_r^i(t+1) = \min\left(p_{min},\ \alpha\frac{r_s^i}{\sum_i r_s^i} + (1-\alpha)p_r^i(t)\right), \quad (4)$$

where $r_s^i = \frac{n_s^i}{n_{\text{off}}}$ (n_s^i and n_{off} are the number of offspring created by i-th operator survived in generation t and total number of offspring survived, respectively). It is possible that at any point during the optimization, no solution generated by one of the repair operators survive. This may cause the corresponding selection probability to go down to zero without any possibility of recovery. To prevent this, in Equation 4, the probability update step ensures that a minimum selection probability (p_{min}) is always assigned to each repair operator present in the ensemble. This ensures that every repair operator has a chance to be selected for generating new solutions.

IV. Truss Design Problem

A. Problem background and formulation

Truss structures provide a wide variety of optimization applications [37], [38], hence, a number of algorithms have been developed in literature to solve them. Genetic algorithms (GAs) have been applied to truss design problems in [39], [40] for single objectives, [41], [42] for multiple objectives, and [43] for many objectives.

For this paper, a scalable truss design problem has been designed based on [44], which makes it suitable to be used as a practical benchmark problem for our proposed algorithms. One such truss having 260 cylindrical members is shown in Fig. 2. It is simply supported at the extreme nodes in the lower portion. All members parallel to the x and y axes are 4 meters long. A vertical load of 10 kN is applied in the negative z-direction to all the top nodes.

The truss design problem can be defined as a bi-objective optimization problem, with the objectives being minimized: (a) weight, and (b) compliance. There are two types of design variables: size variables, defined by member radii ($\mathbf{r}$), and shape variables, defined by length of the vertical members ($\mathbf{l}$) parallel to the z-axis. This problem can be scaled by adjusting the number of shape variables (n_s). Two constraints are to be satisfied: (a) stress at each member should be less than 200 MPa, (b) displacement of each node should be less than 20 mm. The problem formulation is shown below:

$$\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{r},\mathbf{l}) = \rho\sum_{i=1}^{n_m} V_i(\mathbf{r},\mathbf{l}), && (5)\\
\text{Minimize} \quad & f_2(\mathbf{r},\mathbf{l}) = \ \mathbf{F}(\mathbf{r},\mathbf{l})\cdot\mathbf{U}(\mathbf{r},\mathbf{l}), && (6)\\
\text{Subject to} \quad & \sigma_i(\mathbf{r},\mathbf{l}) \le S, \quad \text{for } i = 1,2\ldots,n_m, && (7)\\
& |U_j(\mathbf{r},\mathbf{l})| \le \delta, \quad \text{for } j = 1,2,\ldots,n_n, && (8)\\
& r^L \le r_i \le r^U, \ \text{for } i = 1,2\ldots,n_m, && (9)\\
& l^L \le l_j \le l^U, \ \text{for } j = 1,2\ldots,n_n, && (10)
\end{aligned}$$

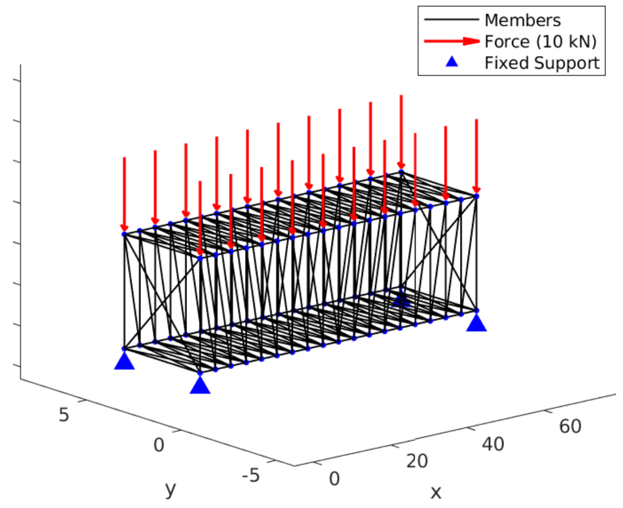


Fig. 2: A truss structure with support and vertical loadings.

where V_i is the volume of the i^{th} member, ρ is the material density (7,000 kg/m^3), n_m is the total number of members, n_n is the total number of nodes, $\mathbf{F}$ and $\mathbf{U}$ are vectors consisting of the forces in members and displacements at nodes, respectively. σ_i is the stress developed i-th member in the truss and is restricted to material strength $S = 200$ MPa. Radius and length variables are restricted to [5, 300] mm and [4, 30] m, respectively. The absolute deflection of nodes is restricted to $\delta = 20$ mm. While f_1 is relatively easy to compute, f_2 and two constraint evaluation require a finite element simulation, thereby making the process computationally expensive with more elements and nodes. Two different trusses are considered here: one with 820 members and 236 nodes, and the other consisting of 1,380 members and 396 nodes. Due to the involvement of a large number of decision variables ($n_s + n_m$), this problem can be considered as an LSMOP.

B. User Knowledge

Truss design problems are very common in the literature and an engineer can provide some expert knowledge about good truss shapes or weight distribution for a particular type of loading. This, in turn, can be mathematically expressed and integrated into the MOEA/I algorithm. Two types of knowledge can be specified for this problem:

- Symmetry - Due to the symmetric nature of the loading and supports, it can be expected that an optimal truss design will be symmetric in terms of shape and weight distribution. For the truss shown in Fig. 2, there can be two planes of symmetry: (a) a plane parallel to the y-z plane and passing through the midpoint of the truss in the x-direction, and, (b) a plane parallel to the x-z plane and passing through the midpoint of the truss in the y-direction.
- Monotonicity - For an optimal truss, length of vertical members will monotonically increase as we move from the support to the middle. In addition, for a simply-supported truss like the one considered here, it is known

that the bending moment monotonically increases from the support towards the middle. So the radius of the corresponding members may also monotonically increase to withstand the large bending moment.

The incorporation of this apriori user knowledge into the optimization process requires defining the different variable groups (G) and the relationship matrix ($\mathbf{U}$) defining the associated relationships. We consider four groups of variables, as shown in Table II. Each group is also divided into two sub-groups. These will be used in specifying symmetry relationships. Each group of size variables is designed taking into account the physical location of the members corresponding to those variables. For example, the members lying parallel to the x-axis at the top of the truss can be expected to be related to each other in some manner rather than a member with a different location and physical orientation.

Based on the variable groups, we have created eight different scenarios (S_1 to S_8) with varying extents of user knowledge being supplied as heuristics, shown in Table III. Each scenario uses the available groups and sub-groups in different ways. Each entry in the table represents the values (u_{ij}) to be filled in the matrix $\mathbf{U}$ for every variable pair $(i, j) \in G_k$. The first four scenarios from S_1 to S_4 do not enforce any symmetry. This makes the optimization more challenging. The scenarios S_5 to S_8 enforce symmetry ($u_{ij} = 4$) in the truss.

In scenario S_1, no knowledge is provided by the user. This is the reference scenario based on which the effectiveness of varying degrees of user knowledge coupled with different innovization-based repair operators is evaluated.

In scenario S_2, only group G_1 is used. u_{ij} is set to 1 for all variable pairs within two sub-groups G_{11} and G_{12}. This signifies that we are expecting the length of each vertical member to consistently follow a monotonically increasing or decreasing pattern, but it is not known beforehand what the exact relationship would be. It is up to the innovization process to determine the exact nature of the relationships among these two groups of variables dictated by the ND solutions.

S_3 extends S_2 and applies $u_{ij} = 1$ among all variables within each sub-group in all groups. Thus, relationships in both shape and size variables will now be obtained by the innovization process and will be enforced by our proposed procedure at various degrees dictated by the three repair schemes.

Scenario S_4 provides more problem information to the optimization algorithm by specifying precise relationships among variables of each sub-group of four groups. For example, within the sub-group G_{11}, variable pairs (x_i-x_{i+1}) are assigned a relationship ($u_{i,i+1} = 2$): $x_i \leq x_{i+1}$ for $i = 1$ to $|G_{11}| - 1$. For G_{12}, $x_i \geq x_{i+1}$ $u_{i,i+1} = 3$) for $i = 1$ to $|G_{12}| - 1$ is assigned.

Scenarios S_5 to S_8 use the same grouping as scenarios S_1 to S_4, respectively. The only addition is the application of symmetry relations within the respective variables of sub-groups in each group. Two planes of symmetry exist as mentioned earlier, and for the corresponding variable pairs (i, j), u_{ij} is set as 4.

Scenarios S_4 and S_8 explicitly define the relationships to be imposed upon on any newly generated solution, and thus, the role of the repair operators is limited to enforcing the user-defined constraints. Thus, it is necessary to compare the performance of the proposed repair operators with methods that handle this type of scenario as constraints. Since, many of the explicitly specified relationships are monotonic in nature, we can use methods such as semi-independent variables (SIVs, v_i) [27] to enforce the provided knowledge as constraints. We call this algorithm NSGA-II/SIV. For example, for sub-group G_{11} with K_{11} variables under scenario S_4 or S_8 defined in Table II, the SIV representation is shown below. A similar process can be repeated for the other variable groups as well.

Desired relationship: $l_1 \leq l_2 \leq \ldots \leq l_{K_{11}}$,
SIVs: l_1 (base var.), v_2, v_3, …, $v_{K_{11}}$ (derived vars.),
where

$$l_i = l_{i-1} + v_i(l^U - l_{i-1}), \;\; i = 2, 3, \ldots, K_{11},$$
$$l^L \leq l_1 \leq l^U,$$
$$0 \leq (v_2, v_3, \ldots, v_{K_{11}}) \leq 1.$$

The restriction of $v_i \leq 1$ ensures that all $u_{i,i+1} = 2$ relationships are always met with the SIVs. Similar updates can be made for $u_{i,i+1} = 3$.

In this paper, we consider only inequality-based relationships due to the nature of the problems considered. However, it is possible to extend the scope to cover other types of relations (such as power laws) as well. In order to test the robustness of the proposed repair operators, different knowledge levels are considered. Experiments are performed to determine how the algorithm performs in the most adverse conditions, such as too little or too much information, and scenarios involving asymmetric trusses. With every scenario, the user knowledge extent is increased, all the way till the final scenario which specifies the exact relationships expected to be present in optimal solutions.

C. Experimental Settings

In this paper, NSGA-II [45], a state-of-the-art MOEA, has been used for the MOEA/I framework. The original NSGA-II algorithm, without any modifications, will be referred to as the base optimization case or base NSGA-II. The 4 types of repair methods, MOEA/IR1, MOEA/IR2, MOEA/IR3 and MOEA/I-ES, presented in Sections III-C to III-F, were used, and are referred to as NSGA-II/IR1, NSGA-II/IR2, NSGA-II/IR3 and NSGA-II/I-ES, respectively. For scenarios S_4 and S_8, additional experiments were performed using semi-independent variables, referred to as NSGA-II/SIV. The parameter settings for NSGA-II as well as the repair operators are presented in Table IV. Two truss cases, one with 820 members, and another with 1,380 members, were considered. The variable groups were set according to Table II. Different experiments were performed with multiple levels of user knowledge integration which are described in Section IV-B. The number of decision variables for scenarios S_1-S_8 are 879 and 1,479 for the 820 and 1,380-member trusses, respectively. For scenarios S_5 to S_8, base NSGA-II takes into account the symmetry relations by evolving only one of any pair of variables following a symmetry relation. Thus, the problems are run with reduced dimensions for these cases only for base NSGA-II. For the 820 member truss, the number of decision variables in that

TABLE II: Variable groups for the 820 and 1,380-member truss cases.

Group	Sub-group	Variable Type	Variable Indices	
			820-member truss	1,380-member truss
G_1	G_{11}	l_i of vertical members	$[1-30]$	$[1-50]$
	G_{12}		$[31-59]$	$[51-99]$
G_2	G_{21}	r_i of top longitudinal members	$[60-117]$	$[60-159]$
	G_{22}		$[118-175]$	$[160-255]$
G_3	G_{31}	r_i of bottom longitudinal members	$[176-233]$	$[256-353]$
	G_{32}		$[234-291]$	$[354-451]$
G_4	G_{41}	r_i of vertical members	$[292-350]$	$[452-550]$
	G_{42}		$[351-409]$	$[551-649]$

TABLE III: Relationship matrix entries (u_{ij}) and groups used for all scenarios. For $u_{ij} = 1$, 2 and 3, the relations are restricted within a subgroup, and for $u_{ij} = 4$, the relation is between the two subgroups.

Group	Sub-group	Scenario							
		S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
G_1	G_{11} G_{12}	0	$\begin{smallmatrix}1\\1\end{smallmatrix}$	$\begin{smallmatrix}1\\1\end{smallmatrix}$	$\begin{smallmatrix}2\\3\end{smallmatrix}$	4	$\begin{smallmatrix}1\\1\end{smallmatrix}4$	$\begin{smallmatrix}1\\1\end{smallmatrix}4$	$\begin{smallmatrix}2\\3\end{smallmatrix}4$
G_2	G_{21} G_{22}	0	0	$\begin{smallmatrix}1\\1\end{smallmatrix}$	$\begin{smallmatrix}2\\3\end{smallmatrix}$	0	0	$\begin{smallmatrix}1\\1\end{smallmatrix}4$	$\begin{smallmatrix}2\\3\end{smallmatrix}4$
G_3	G_{31} G_{32}	0	0	$\begin{smallmatrix}1\\1\end{smallmatrix}$	$\begin{smallmatrix}2\\3\end{smallmatrix}$	0	0	$\begin{smallmatrix}1\\1\end{smallmatrix}4$	$\begin{smallmatrix}2\\3\end{smallmatrix}4$
G_4	G_{41} G_{42}	0	0	$\begin{smallmatrix}1\\1\end{smallmatrix}$	$\begin{smallmatrix}2\\3\end{smallmatrix}$	0	0	$\begin{smallmatrix}1\\1\end{smallmatrix}4$	$\begin{smallmatrix}2\\3\end{smallmatrix}4$

case would be 850 for S_5-S_6 (shape symmetry) and 450 for S_7-S_8 (shape and size symmetry). For the 1380 member truss, the number of decision variables in that case would be 1439 for S_5-S_6 and 750 for S_7-S_8. 20 runs were performed and the mean and standard deviation of every scenario and repair operator is reported. For both truss cases, the maximum number of computations available was set at 2 million.

Hypervolume (HV) metric [46], [47] is used as a performance metric. The median HV of scenario S_1 achieved by base NSGA-II is set as the target hypervolume (HV^T). Performance comparison is performed by measuring the average number of function evaluations taken by each method to achieve HV^T.

Wilcoxon rank-sum test [48], [49] 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 x_1 and x_2 represent the final HV values for two algorithms A_1 and A_2. For each run, we have x_1 and x_2 as paired observations. Here, the null hypothesis states that the there is no statistically significant difference between x_1 and x_2. The hypothesis was tested with 95% significance level and the p-values are recorded. A p-value > 0.05 means that there is a statistically insignificant difference between the best and another algorithm. In this case, the algorithm with the lowest number of median function evaluations (FE$^{\text{min}}_{\text{median}}$) required for achieving HV^T is chosen as reference. The HV values for achieving one standard deviation more than lowest median FE (FE$^{\text{min}}_{\text{median}}$+$\sigma^{\text{min}}_{\text{FE}}$) for all algorithms are recorded for 20 runs. Using this data, the Wilcoxon test is performed and the p-values are calculated.

TABLE IV: Parameter settings of NSGA-II/I.

Parameter	Value
Population size	500
Maximum generations	4,000
Mutation operator	Polynomial mutation [50]
Mutation probability (p_m) and index (η_m)	$1/n_{var}$, 50
Crossover operator	SBX [25]
Crossover probability (p_c) and index (η_c)	0.9, 30
Momentum parameter (γ)	0.2
Minimum repair prob. (p_{min}) in Eqn. 4	0.1
α in Equation 4	0.5

D. Results and Discussion

The optimization results for the 820 and 1,380-member truss cases are presented Tables V and VI, respectively. Scenarios S_1 to S_8 are designed to show the results of the proposed MOEA/I approach (implemented as NSGA-II/I), and also for NSGA-II paired with SIVs. In both the truss cases, the row for scenario S_1 is marked as N/A for all the algorithms except for NSGA-II. This is because S_1 is the no-knowledge case where all the elements in matrix $\mathbf{U}$ are set at 0. Thus, all the repair operators remain inactive, and the results are the same as base NSGA-II. For most of the other cases it is seen that NSGA-II/IR2 performs the best (marked in bold), except for scenario S_8. This shows that a moderate level of knowledge obtained through innovization gives the optimal results. Too little or too much knowledge is detrimental to the optimization performance. NSGA-II/I-ES has a slightly worse performance than NSGA-II/IR2 in general, but it is still better overall than IR1 and IR2. It also has the advantage that the user does not need to think about which repair operator to use since it is handled automatically.

If we proceed row-wise from top to bottom, it is seen that performance generally improves till a particular point (S_3 and S_7) and then drops (S_4 and S_8). Interestingly, S_4 and S_8 are the ones which explicitly specify the relationships instead of letting the IR operators determine it. A possible cause is that over-specification of knowledge complicates the search process and constrains the algorithm efficiency. This can be verified by the results of NSGA-II/SIV, where no repair operators are used and the knowledge is effectively treated as a constraint. The results given by the SIV representations are similar to that of S_4 and S_8. Results of the Wilcoxon test are given in braces in each cell. Except for scenarios S_4 and S_8, all p-values come out to be less than 0.05 which means statistically, the repair operators perform better than the Base

TABLE V: FEs (median and standard deviation) required to reach target HV^T = 0.91 for 820-member, 236-node truss problem. Best performing algorithm is marked in bold. Algorithms performing in statistically insignificant manner to the best algorithm are marked in bold italics. N/A indicates 'Not Applied' due to similarity with the base algorithm or not applicable with the SIV concept.

Scenario	$\mathcal{K}$	Algorithm Used					
		Base NSGA-II	NSGA-II/IR1	NSGA-II/IR2	NSGA-II/IR3	NSGA-II/I-ES	NSGA-II/SIV
S_1	0	2M	N/A	N/A	N/A	N/A	N/A
S_2	7.63e-05	2M ($p = 0.0301$)	***1.6M*** $\pm$ 65.0k ($p = 0.1343$)	**1.4M** $\pm$ 72.0k	***1.7M*** $\pm$ 9.0k ($p = 0.0951$)	***1.5M*** $\pm$ 50.0k ($p = 0.1520$)	N/A
S_3	3.80e-04	2M ($p = 0.0213$)	***1.2M*** $\pm$ 30.0k ($p = 0.1830$)	**0.98M** $\pm$ 33.0k	***1.5M*** $\pm$ 25.0k ($p = 0.0961$)	***1.2M*** $\pm$ 27.0k ($p = 0.1841$)	N/A
S_4	5.60e-04	***2M*** ($p = 0.0624$)	1.9M $\pm$ 23.0k ($p = 0.0411$)	**1.7M** $\pm$ 20.0k	2M (HV=0.52, $p = 0.0068$)	***1.8M*** $\pm$ 10.0k ($p = 0.1313$)	***1.9M*** $\pm$ 20.0k ($p = 0.0513$)
S_5	6.50e-04	**1.2M** $\pm$ 8.0k	***1.3M*** $\pm$ 18.0k ($p = 0.1363$)	***1.2M*** $\pm$ 12.0k ($p = 0.1413$)	***1.6M*** $\pm$ 8.0k ($p = 0.1526$)	***1.3M*** $\pm$ 21.0k ($p = 0.1201$)	N/A
S_6	7.60e-04	***1.2M*** $\pm$ 8.0k ($p = 0.0714$)	***0.99M*** $\pm$ 22.0k ($p = 0.1328$)	**0.96M** $\pm$ 25.0k	***1.2M*** $\pm$ 22.0k ($p = 0.0983$)	***0.98M*** $\pm$ 21.0k ($p = 0.1154$)	N/A
S_7	1.16e-03	1M $\pm$ 10.0k ($p = 0.0048$)	***0.72M*** $\pm$ 19.0k ($p = 0.0612$)	**0.44M** $\pm$ 23.0k	0.92M $\pm$ 10.0k ($p = 0.0141$)	***0.65M*** $\pm$ 15.0k ($p = 0.0596$)	N/A
S_8	1.30e-03	**1M** $\pm$ 10.0k	***1.3M*** $\pm$ 21.0k ($p = 0.0662$)	1.6M $\pm$ 30.0k ($p = 0.0419$)	1.8M $\pm$ 20.0k ($p = 0.0261$)	1.5M $\pm$ 27.0k ($p = 0.0237$)	1.8M $\pm$ 10.0k ($p = 0.0225$)

TABLE VI: FEs required to reach target HV^T = 0.80 for 1,380-member, 396-node truss problem.

Scenario	$\mathcal{K}$	Algorithm Used					
		Base NSGA-II	NSGA-II/IR1	NSGA-II/IR2	NSGA-II/IR3	NSGA-II/I-ES	NSGA-II/SIV
S_1	0	2M	N/A	N/A	N/A	N/A	N/A
S_2	3.14e-03	***2M*** ($p = 0.2347$)	***1.8M*** $\pm$ 42.0k ($p = 0.3212$)	**1.6M** $\pm$ 48.0k	***1.8M*** $\pm$ 9.0k ($p = 0.3205$)	***1.7M*** $\pm$ 25.0k ($p = 0.3153$)	N/A
S_3	4.70e-03	2M ($p = 0.0153$)	***1.6M*** $\pm$ 20.0k ($p = 0.1394$)	**1.4M** $\pm$ 25.0k	1.9M $\pm$ 10.0k ($p = 0.0265$)	***1.5M*** $\pm$ 17.0k ($p = 0.1277$)	N/A
S_4	6.27e-03	**2M**	2M (HV=0.31, $p = 0.0019$)	2M (HV=0.44, $p = 0.0058$)	2M (HV=0.57, $p = 0.0103$)	2M (HV=0.66), $p = 0.0216$)	2M (HV=0.30, $p = 0.0038$)
S_5	6.52e-03	**1.6M** $\pm$ 25.0k	***1.7M*** $\pm$ 15.0k ($p = 0.3120$)	***1.7M*** $\pm$ 23.0k ($p = 0.2961$)	***1.8M*** $\pm$ 12.0k ($p = 0.1849$)	***1.7M*** $\pm$ 11.0k ($p = 0.2971$)	N/A
S_6	3.26e-02	1.6M $\pm$ 25.0k ($p = 0.0431$)	***1.3M*** $\pm$ 20.0k ($p = 0.0713$)	**1.0M** $\pm$ 30.0k	1.5M $\pm$ 10.0k ($p = 0.0481$)	***1.1M*** $\pm$ 18.0k ($p = 0.0692$)	N/A
S_7	3.59e-02	1.4M $\pm$ 15.0k ($p = 0.0303$)	***1.2M*** $\pm$ 19.0k ($p = 0.0572$)	**0.90M** $\pm$ 23.0k	***1.2M*** $\pm$ 10.0k ($p = 0.0576$)	***1.0M*** $\pm$ 15.0k ($p = 0.0861$)	N/A
S_8	4.83e-02	**1.4M** $\pm$ 15.0k	2M (HV=0.68, $p = 0.0422$)	2M (HV=0.73, $p = 0.0490$)	2M (HV=0.72, $p = 0.0481$)	2M (HV=0.71, $p = 0.0477$)	2M (HV=0.66, $p = 0.0302$)

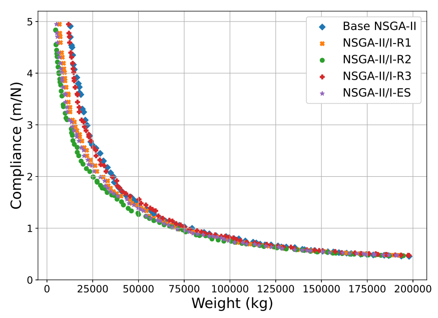
NSGA-II on the truss problems with 95% confidence. Another reason for superior performance of Base NSGA-II in S_4 to S_8 is the reduced number of variables, as all pairs with $u_{ij} = 4$ are assumed equal.

Results for both trusses for scenario S_7 are shown in Figs. 3 and 4, respectively. The ND front plot clearly shows NSGA-II/IR2 as the better performer, providing higher quality solutions than the others, with NSGA-II/I-ES following close behind. The median HV plot also shows that NSGA-II/IR2 and NSGA-II/I-ES achieves almost the same level of performance of the no-knowledge case in much fewer iterations as well as gives better performance. The median repair probabilities plot shows the selection probability of each repair operator as well as base NSGA-II for generating new solutions in NSGA-II/I-ES. It is seen that in both truss cases, a high probability is assigned to IR2 compared to others. Probabilities of IR1 and IR3 are reduced to the minimum level very early on, and new solution generation is controlled mostly by IR2 and base NSGA-II. This clearly indicates the ability of the proposed ensembled method (I-ES) to pick the most successful operator for solving a problem efficiently.

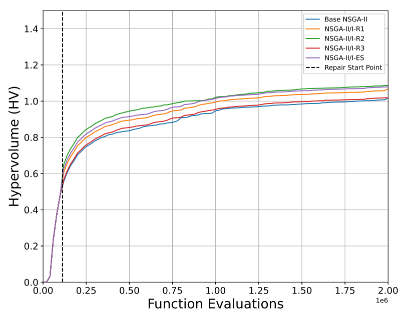
For all the NSGA-II/IR algorithms, performance always degrades when knowledge is explicitly specified. Such exact information seems to have a constraining effect on the search process. These algorithms perform best if they are provided with some basic guidance and then left to discover the underlying relationships on their own.

An interesting comparison is the difference in performances between NSGA-II/IR approaches and NSGA-II/SIV (applicable to cases with $u_{ij} = 2$ or 3 only). For the 820-member truss case in Table V, the performance of NSGA-II/SIV is close to NSGA-II/I-ES and inferior to NSGA-II/IR2 for S_4. For S_8, NSGA-II/SIV is notably worse than the NSGA-II/IR approaches. A greater amount of supplied knowledge seems to provide only a marginal improvement in the performance of NSGA-II/SIV. For the 1,380-member truss, all the algorithms are unable to reach the desired performance within 2M function evaluations. This shows that SIV approaches are still susceptible to knowledge overspecification.

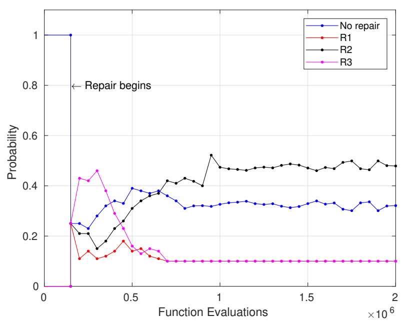
It is interesting from Tables V and VI that with more information being provided $\mathcal{K}$ the performance of all algorithms does not improve monotonically. This means that there exists an optimal amount of problem information for every problem below or above which there is under or over-specification of



(a) ND fronts.

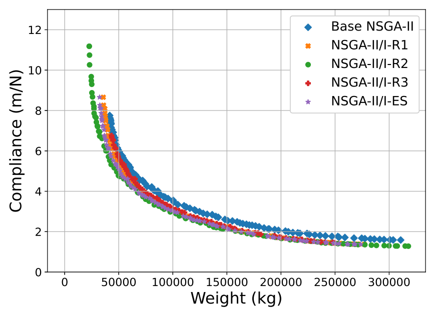


(b) Median hypervolume of 20 runs.

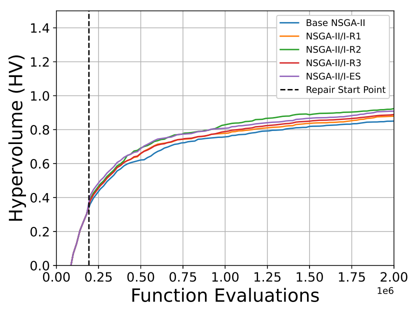


(c) Repair prob. variation for median run.

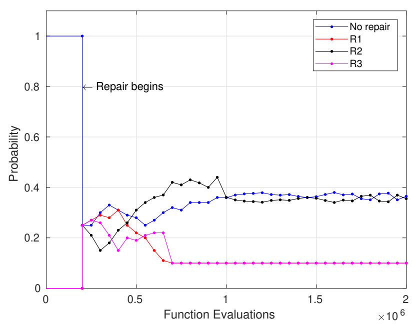
Fig. 3: Results for 820-member truss for scenario S_7 (symmetry among sub-groups, unknown relationships within sub-groups).



(a) ND fronts.



(b) Median hypervolume of 20 runs.



(c) Repair prob. variation for median run.

Fig. 4: Results for 1,380-member truss for scenario S_7 (symmetry among sub-groups, unknown relationships within sub-groups).

information provided. When less than necessary information is provided, an algorithm needs to work its way to find relevant building blocks for solving the problem thereby requiring more solution evaluations. On the other hand, if more than necessary information is provided, even if the information is correct, the algorithm may get restricted to discover other required information needed to solve the problem. Providing just an optimal amount of additional problem information and without unnecessary restriction in creating novel solutions enables an EA to an optimal number of solution evaluations to solve a problem. For both 820 and 1,380 member problems, we observe that S_7 scenario puts the right amount of additional problem information for NSGA-II/IR2 (repair 2) algorithm to take the right amount of advantage to solve he problem with at most 22% and 45% of median solution evaluations than the Base NSGA-II procedure.

V. Solid Rocket Design Problem

A. Problem background and formulation

Solid rocket fuels have seen an increased usage over many years due to the advantages like lower cost and simpler designs [51]. Optimal designs of solid rocket motors is thus a pertinent problem to NASA and other private organizations. The main concern in designing a solid rocket motor is with choosing a suitable propellant composition, thickness of layers, and core propellant geometry which can meet a set of pre-determined performance criteria. This can be formulated as an optmization problem [52]. There are two criteria a good design must satisfy: (a) it should be able to meet a certain target thrust profile (thrust versus time), and (b) it should minimize the amount of insulation required in the motor by ensuring the entirety of the propellant gets burnt out simultaneously. Savings in insulation weight allows for an increase in payload. In traditional solid rocket propellants the composition is more or less uniform. In this case, however, the composite propellant is made up of multiple sub-regions of same or different propellant types, each with a separate burning rate, and following a specific geometry. The optimization algorithm has to find a design that specifies which propellant to use and how they are arranged. Current manufacturing technologies face a lot of challenge in implementing such complex motor designs. However, with advances in 3-D printing technology, eventually, it may be become feasible to manufacture such designs.

In this paper, we have adopted the optimization problem formulation and burn simulation model given in [52]. The rocket under consideration has the propellant regions divided vertically into six cylindrical sections, including the dome, with each having a maximum of 20 distinct layers arranged as concentric rings. Three sections possess some additional propellant arranged at the core, also referred to as a ”star” shape. In addition, there are six smaller non-cylindrical ”corner” segments acting as an interface between the dome and cylindrical segment propellants. Lastly, the nozzle region is also considered a separate segment with different propellant layers.

The target thrust profile is defined as a monotonically decreasing curve defined at 0.5 second intervals, and lasting for 10 seconds, as shown by the blue band in Fig. 6a. A good design should match the target thrust profile with a tolerance of $\pm 5\%$. The allowable pressure range is set to be within 1.4 MPa to 3.5 MPa. The nozzle region has a pre-specified insulation geometry for proper burning. 11 propellant types are allowed for this problem, numbered in increasing order of burn rate, as shown in Table VII. Instead of using the exact reference burn rates, the decision variables pertaining to the propellant choice will be replaced by the values under the 'Type' column in Table VII.

TABLE VII: Available propellant types and their burn rates.

Type	Reference Burn Rate (m/s)
0	2.54e-03
1	3.05e-03
2	3.63e-03
3	4.34e-03
4	5.21e-03
5	6.22e-03
6	7.44e-03
7	8.92e-03
8	1.064e-02
9	1.275e-02
10	1.524e-02

The optimization problem formulation is given below.

$$\text{Minimize} \quad f_1(\mathbf{x}) = \sum_{t=0}^{t_b} (T(\mathbf{x}, t) - T_r(t))^2, \qquad (11)$$

$$\text{Minimize} \quad f_2(\mathbf{x}) = \mu_{res}(\mathbf{x}) + \sigma_{res}(\mathbf{x}), \qquad (12)$$

$$\text{subject to} \quad P^L \leq P(\mathbf{x}, t) \leq P^U, \qquad (13)$$

where $\mathbf{x}$ is the vector of decision variables comprising of propellant distribution along segments of each layer, thickness of each layer and geometry of core propellant arrangement. $T(\mathbf{x}, t)$ is the thrust obtained at time t, t_b is the target burn time, $T_r(t)$ is the target thrust at time t, $\mu_{res}(\mathbf{x})$ and $\sigma_{res}(\mathbf{x})$ are the mean and standard deviation of segment residues, respectively, $P(\mathbf{x}, t)$ is the pressure at time t, P^L and P^U are the minimum and maximum allowable pressures, respectively.

There are both discrete and continuous variables, which add to the problem complexity. In addition, the number of decision variables (544) is also high, making it an LSMOP. The breakdown of each type of decision variables is given in Table VIII. The user can set the cylindrical layer propellants and thicknesses, the geometry of the star sections as well as their propellant composition.

TABLE VIII: Decision variable ($\mathbf{x}$) properties for the rocket problem.

Variables	Type	Range	Number
Layer propellants	Discrete	[0, 10]	260
Layer thicknesses	Continuous	[1, 2.35]	260
Star geometry	Discrete	[0, 3]	18
Star propellants	Discrete	[0, 35]	3
Circularization propellant	Discrete	[0, 10]	3

The first objective function defined by Eq. (11) measures the sum of squared errors (SSE) between the target thrust and that obtained by the current design. The error at time t is considered as 0 if the obtained thrust is within $\pm 5\%$ of the target thrust.

The second objective function defined by Eq. (12) is an indirect measure of the amount of insulation required and the extent of simultaneous burnout. Here, residue is measured by the thickness of the cylindrical fuel layer (in metres) remaining in each segment at the end of the burn, when the propellant in atleast one of the segments has fully burnt out. The mean (μ_{res}) of the residues at all the segments aims to encourage the maximum propellant usage and lengthen the burn period. The standard deviation (σ_{res}) of all the segment residues measures the disparity in the residues across different segments. Minimizing both μ_{res} and σ_{res} is necessary for obtaining a good design.

Equation (13) represents the pressure constraint which is necessary to ensure optimal rocket operation. Dropping below the minimum pressure will cause the rocket to irrevocably lose thrust, whereas, exceeding the pressure risks explosion of the rocket.

B. User Knowledge

Unlike the truss design problem which is very well-studied in literature, the solid rocket design problem is very new. This problem is intended to demonstrate the effectiveness of our proposed approach in a new and less-analyzed practical problem. Due to the lack of previous knowledge, most of the supplied user knowledge cases analyzed here will not specify any exact relationships, but rather, leave it to the innovization process to figure them out. A logical way to define variable groups is to do it segment-wise, as shown in Table IX. Groups G_{r1}-G_{r3} represent the star shape for the 3 star segments. G_{r4}-G_{r9} represent the propellants for each cylindrical segment. G_{r10}-G_{r15} represent the layer thicknesses for each cylindrical segments.

Five scenarios can be designed based on different levels of user knowledge extent, shown in Table X. Scenario C_1 represents the no-knowledge case of the optimization. This is the reference scenario and the optimization is free to evolve all 544 variables in any fashion.

Scenario C_2 is based on the knowledge that star segments determine a large proportion of the initial part of the burn [52]. As a result, specific patterns might result in a better rocket performance. Thus, for groups G_{r1}-G_{r3}, u_{ij} is set as 1.

Scenario C_3 extends C_2 and by including the propellant variables for each cylindrical segment (groups G_{r4}-G_{r9}) into the scope of the innovization. Thus, any pattern that exists between the propellants in each segment will be captured, and repair performed accordingly.

Scenario C_4 builds upon C_3 and also includes the layer thickness variables (groups G_{r10}-G_{r15}). This scenario uses all the groups defined in Table IX.

Scenario C_5 is intended to demonstrate how more knowledge may not necessarily work in the favor of the optimization algorithm. In this case, the nature of the target thrust profile is considered, and an explicit relation among the cylindrical layer propellants are supplied. The target thrust profile in Fig. 6a is decreasing in nature, and the burn progresses outwards from the core to the shell. Hence, an argument can be made that the near-core layers should have a faster burning fuel, and the far-core layers should have slower burning fuel. This can be expressed by setting $u_{ij} = 3$ for every group G_{r4}-G_{r9}. Compared to the other scenarios, C_5 consists of the maximum user-supplied information.

TABLE IX: Variable groups for the rocket problem.

Variable Type	Group	Variable Indices
Star segment geometry	G_{r1}	$[521-526]$
	G_{r2}	$[529-534]$
	G_{r3}	$[537-542]$
Cylindrical segment propellants	G_{r4}	$[1-20]$
	G_{r5}	$[21-40]$
	G_{r6}	$[41-60]$
	G_{r7}	$[61-80]$
	G_{r8}	$[81-100]$
	G_{r9}	$[101-120]$
Cylindrical segment layer thickness	G_{r10}	$[261-280]$
	G_{r11}	$[281-300]$
	G_{r12}	$[301-320]$
	G_{r13}	$[321-340]$
	G_{r14}	$[341-360]$
	G_{r15}	$[361-380]$

TABLE X: Relationship matrix entries (u_{ij}) and groups used for all scenarios for the rocket problem. More problem information is provided with increasing index in C.

Group	Scenario				
	C_1	C_2	C_3	C_4	C_5
G_{r1}	0	1	1	1	1
G_{r2}	0	1	1	1	1
G_{r3}	0	1	1	1	1
G_{r4}	0	0	1	1	3
G_{r5}	0	0	1	1	3
G_{r6}	0	0	1	1	3
G_{r7}	0	0	1	1	3
G_{r8}	0	0	1	1	3
G_{r9}	0	0	1	1	3
G_{r10}	0	0	0	1	1
G_{r11}	0	0	0	1	1
G_{r12}	0	0	0	1	1
G_{r13}	0	0	0	1	1
G_{r14}	0	0	0	1	1
G_{r15}	0	0	0	1	1

C. Experimental Settings

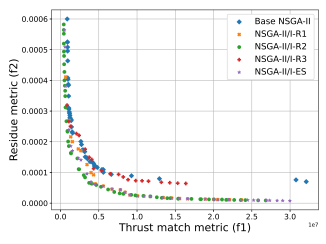
As was the case for the truss problem, NSGA-II was used for the MOEA/I framework. A total of five algorithms, Base NSGA-II, NSGA-II/IR1, NSGA-II/IR2, NSGA-II/IR3 and NSGA-II/I-ES, respectively, were applied. For scenario C_5 consisting of explicitly-specified information, NSGA-II/SIV is used. The parameter settings for NSGA-II as well as the repair operators are presented in Table IV. Different experiments are performed with multiple levels of user knowledge integration, shown in Table X. HV metric is used to measure the convergence and diversity of the solutions generated by each method. Statistical performance is measured using the Wilcoxon Rank Sum Test.

D. Results and Discussion

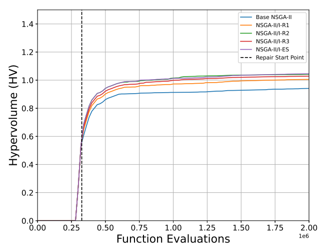
The optimization results are presented Table XI. The extent of supplied information increases row-wise from scenario C_1 to C_5 as can be seen from the values of $\mathcal{K}$. For the no-knowledge case C_1, all the elements of matrix $\mathbf{U}$ are set as 0. Thus, all repair operators remain inactive, so apart from base NSGA-II, all the other cells are marked as N/A. It is seen that in all of the cases except for C_5, NSGA-II/IR2 is the best performing algorithm (marked in bold), with NSGA-II/I-ES showing a comparable performance. For all the scenarios, the p-values of the Wilcoxon test with the best-performing algorithm as a reference are also given in brackets. Scenario C_4 combined with the NSGA-II/R2 operator gives the best performance overall. This shows that for both the amount of user-supplied knowledge and the extent of knowledge usage, an optimal level exists. Too little or too much knowledge usage is detrimental to the optimization performance. Interestingly, for C_5 we explicitly try to define the relationships the propellants in the cylindrical layers should follow. But this seems to constrain the algorithm and degrades their performance to below that of base NSGA-II. Since this problem is not very well-studied, a possible cause could be the lack of a direct relation between the thrust and the individual layer propellants. The same thrust value can be produced by a lot of different propellant combinations across all segments, and simple monotonic relationships may not exist.

In terms of final HV obtained, for all the cases, NSGA-II/IR2 provides the fastest convergence, demonstrated by the low number of function evaluations taken to reach the target HV, with NSGA-II/ES following closely. The final HV for all the repair operators are statistically likely to be similar, as demonstrated by the high p-values (> 0.05). However, the Base NSGA-II always performs worse in scenarios C_2 to C_4. For C_5, the perceived knowledge $u_{ij} = 3$ for G_{r4}-G_{r9} is found to be not correct and they harm the performance of NSGA-II/IR methods.

One of the non-dominated fonts obtained by the four repair operators for scenario C_4 is shown in Fig. 5a. It can be seen that NSGA-II/IR2 provides better quality solutions than the others. The median HV plot in Fig. 5b also shows that NSGA-II/IR2 achieves the same level of performance as that of the no-knowledge case with fewer function evaluations. The thrust profile and residues for one of the solutions on the ND front is shown in Figs. 6a and 6b, respectively. Thrust is always within allowable ranges and the residue for every segment is lower than 1 mm.



(a) ND fronts.

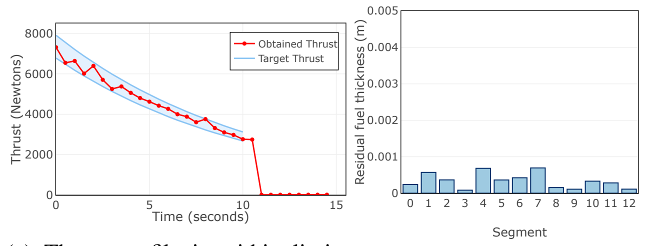


(b) HV variations.

Fig. 5: ND fronts and HV variations of scenario C_4 for rocket design.

TABLE XI: FEs required to reach target HV^T = 0.93 for 544-variable solid fuel rocket design.

Scenario	$\mathcal{K}$	Algorithm Used					
		Base NSGA-II	NSGA-II/IR1	NSGA-II/IR2	NSGA-II/IR3	NSGA-II/I-ES	NSGA-II/SIV
C_1	0	50.0M	N/A	N/A	N/A	N/A	N/A
C_2	1.5e-04	50.0M ($p = 0.0053$)	***32.0M*** $\pm$ 2.0M ($p = 0.1205$)	**30.5M** $\pm$ 1.5M	***33.0M*** $\pm$ 3.0M ($p = 0.1044$)	***31.5M*** $\pm$ 1.0M ($p = 0.1516$)	N/A
C_3	4.0e-03	50.0M ($p = 0.0061$)	***28.0M*** $\pm$ 3.0M ($p = 0.0998$)	**25.0M** $\pm$ 4.0M	***32.0M*** $\pm$ 1.0M ($p = 0.1005$)	***26.0M*** $\pm$ 1.0M ($p = 0.1336$)	N/A
C_4	7.8e-03	50.0M ($p = 0.0029$)	***23.0M*** $\pm$ 5.0M ($p = 0.5035$)	**18.0M** $\pm$ 5.0M	***28.0M*** $\pm$ 4.0M ($p = 0.1154$)	***20.0M*** $\pm$ 2.0M ($p = 0.7710$)	N/A
C_5	9.8e-03	**50.0M**	50.0M (HV=0.74, $p = 0.0233$)	50.0M (HV=0.71, $p = 0.0135$)	50.0M (HV=0.66, $p = 0.0104$)	50.0M (HV=0.73, $p = 0.0157$)	50.0M (HV=0.69, $p = 0.0119$)



(a) Thrust profile is within limits (f_1).

(b) Residue (f_2) is less than 1 mm.

Fig. 6: Objectives f_1 and f_2 for a particular run for scenario C_4.

VI. Overall Performance of Proposed Methods

We rank each of the three case studies (879 and 1,479-variable truss designs and 544-variable rocket design) based on the median FEs needed by each of the five algorithms (Base NSGA-II, three innovized repair based NSGA-IIs, and ensembled NSGA-II) in 20 independent runs of each. The sum of ranks of each algorithm over all 3 case studies are calculated, and the algorithms are ranked accordingly. The final row of Table XII indicates that NSGA-II/IR2 performs the best, followed by NSGA-II/I-ES. The Base NSGA-II and NSGA-II/IR1 are tied in the third place. NSGA-II/IR3 is ranked last, showing an aggressive use of available information performs the worst. Interestingly, even though a single balance of problem information and its use within NSGA-II (IR2) performs the best for these problems, with all repair methods being included, our proposed ensembled NSGA-II performs the second best.

TABLE XII: Ranking of different NSGA-IIs on three case studies.

Scenario	Base	IR1	IR2	IR3	ES
879-variable Truss Design					
S_2	5	3	1	4	2
S_3	5	3	1	4	2
S_4	4	3	1	5	2
S_5	1	3	2	5	4
S_6	4	3	1	5	2
S_7	5	3	1	4	2
S_8	1	2	4	5	3
Sum	25	20	11	32	17
Rank	4	3	1	5	2
1,479-variable Truss Design					
S_2	5	4	1	3	2
S_3	5	3	1	4	2
S_4	1	5	4	3	2
S_5	1	3	4	5	2
S_6	5	3	1	4	2
S_7	5	4	1	3	2
S_8	1	5	2	3	4
Sum	23	27	14	25	16
Rank	3	5	1	4	2
544-variable Rocket Design					
C_2	5	3	1	4	2
C_3	5	3	1	4	2
C_4	5	3	1	4	2
C_5	1	2	4	5	3
Sum	16	11	7	17	9
Rank	4	3	1	5	2
Rank-Sum	11	11	3	14	6
Final Rank	**3**	**3**	**1**	**4**	**2**

VII. Conclusions and Future Work

Solving LSMOPs is a challenge for many popular MOEAs. Identifying, learning and applying any hidden patterns among variables of high-performing solutions is a potential way to make MOEAs scalable to solve LSMOPs. In this paper, in addition to exploiting such patterns observed during an optimization process through the “innovization” concept, we have proposed a framework in which guidance from experienced users in terms of additional problem information is sought to enable repair-based algorithms to focus on identified variable combinations. We have proposed three repair operators which utilize the supplied problem information differently with low to high confidence. On two real-world complex problems with 544 to 1,479 variables involving time-consuming evaluation procedures, we have systematically introduced problem information in different scenarios. Moreover, we have proposed an ensemble-based MOEA to adaptively use the three repair operator based on their performance in the past generations. The following observations can be made from our detailed systematic study:

- The amount of additional problem information provided to an algorithm is crucial to achieve the best performance. Too little or too much information is found to be not beneficial.
- Too little or too much trust (or utilization) of the supplied information is also found to be harmful to the search process.
- Since the ideal amount of problem information and its utilization level are not known a priori for a problem, an ensemble-based method is proposed, which is shown to provide a good compromise.
- It is demonstrated that the use of adequate additional

problem information and its proper utilization within an MOEA allows a faster (with only 22% to 45% overall function evaluations in the current tests) compared to the baseline MOEA in solving large-scale problems.

The relationship matrix defined for MOEA/I paves a way to develop a truly interactive optimization algorithm in which additional problem information can be sought at regular intervals. This will not only increase search efficiency, but will also provide useful information to the user with respect to the evolution of good solutions from start to finish. We believe such collaborative efforts are key to enabling the solution of large-scale and complex optimization problems. This study has demonstrated a path towards achieving such an interactive optimization algorithm, but further investigation is needed to include more generic relationships (such as, power laws, periodic relations, etc.) intermittently guided by experienced users and automatically detected and validated by smart and machine learning based optimization algorithms. Nevertheless, this is one of the few studies in which more than 1,000-variable real-world multi-objective optimization problems have been solved with less than 50% computational effort compared to a baseline algorithm.

Acknowledgment

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. We thank Qiren Gao and Ming Liu for their valuable inputs.

References

[1] Z. Yang, K. Tang, and X. Yao, “Large scale evolutionary optimization using cooperative coevolution,” *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, aug 2008.

[2] J. J. Durillo, A. J. Nebro, C. A. Coello, F. Luna, and E. Alba, “A comparative study of the effect of parameter scalability in multi-objective metaheuristics,” in *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, 2008, pp. 1893–1900.

[3] J. J. Durillo, A. J. Nebro, C. A. Coello, J. Garcia-Nieto, F. Luna, and E. Alba, “A Study of multiobjective metaheuristics when solving parameter scalable problems,” *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 618–635, aug 2010.

[4] R. Bellman, “Dynamic Programming Princeton University Press,” *Princeton, NJ*, 1957.

[5] K. Deb and C. Myburgh, “A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables,” *European Journal of Operational Research*, vol. 261, no. 2, pp. 460–474, 2017.

[6] F. Chicano, D. Whitley, R. Tinós, and G. Ochoa, “Optimizing One Million Variable NK Landscapes by Hybridizing Deterministic Recombination with Local Search.” in *Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2017, pp. 753–760.

[7] W. N. Chen, Y. H. Jia, F. Zhao, X. N. Luo, X. D. Jia, and J. Zhang, “A Cooperative co-evolutionary approach to large-scale multisource water distribution network optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 842–857, oct 2019.

[8] Y. Tian, R. Liu, X. Zhang, H. Ma, K. C. Tan, and Y. Jin, “A Multi-Population Evolutionary Algorithm for Solving Large-Scale Multi-Modal Multi-Objective Optimization Problems,” *IEEE Transactions on Evolutionary Computation*, pp. 1–1, dec 2020.

[9] C. He, R. Cheng, C. Zhang, Y. Tian, Q. Chen, and X. Yao, “Evolutionary Large-Scale Multiobjective Optimization for Ratio Error Estimation of Voltage Transformers,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 868–881, oct 2020.

[10] A. Ahrari and K. Deb, “An improved fully stressed design evolution strategy for layout optimization of truss structures,” *Computers and Structures*, vol. 164, pp. 127–144, feb 2016.

[11] K. Wu, J. Liu, X. Hao, P. Liu, and F. Shen, “An Evolutionary Multi-Objective Framework for Complex Network Reconstruction Using Community Structure,” *IEEE Transactions on Evolutionary Computation*, pp. 1–1, aug 2020.

[12] K. Wu, J. Liu, and S. Wang, “Reconstructing networks from profit sequences in evolutionary games via a multiobjective optimization approach with lasso initialization,” *Scientific Reports*, vol. 6, nov 2016.

[13] M. A. Potter and K. A. De Jong, “A cooperative coevolutionary approach to function optimization,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 866 LNCS. Springer Verlag, 1994, pp. 249–257.

[14] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, “A Decision Variable Clustering-Based Evolutionary Algorithm for Large-Scale Many-Objective Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 97–112, feb 2018.

[15] B. Cao, J. Zhao, Z. Lv, and X. Liu, “A Distributed Parallel Cooperative Coevolutionary Multiobjective Evolutionary Algorithm for Large-Scale Optimization,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2030–2038, aug 2017.

[16] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong, “A Multiobjective Evolutionary Algorithm Based on Decision Variable Analyses for Multiobjective Optimization Problems with Large-Scale Variables,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 275–298, apr 2016.

[17] R. Liu, J. Liu, Y. Li, and J. Liu, “A random dynamic grouping based weight optimization framework for large-scale multi-objective optimization problems,” *Swarm and Evolutionary Computation*, vol. 55, p. 100684, jun 2020.

[18] L. M. Antonio and C. A. Coello Coello, “Decomposition-based approach for solving large scale multi-objective problems,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9921 LNCS. Springer Verlag, 2016, pp. 525–534.

[19] L. M. Antonio and C. A. Coello, “Use of cooperative coevolution for solving large scale multiobjective optimization problems,” in *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, 2013, pp. 2758–2765.

[20] W. Hong, K. Tang, A. Zhou, H. Ishibuchi, and X. Yao, “A Scalable Indicator-Based Evolutionary Algorithm for Large-Scale Multiobjective Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 525–537, jun 2019.

[21] 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.

[22] 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.

[23] 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.

[24] Q. Zhang, A. Zhou, and S.-Z. Zhao, “Multiobjective optimization test instances for the CEC 2009 special session and competition,” *University of Essex, . . .*, no. January 2008, pp. 1–30, 2008. [Online]. Available: http://dces.essex.ac.uk/staff/zhang/MOEAcompetition/cec09testproblem0904.pdf.pdf

[25] K. Deb and R. Bhushan Agrawal, “Simulated Binary Crossover for Continuous Search Space Kalya nmoy D eb' Ram B hushan A gr awal,” *Complex Syst.*, vol. 9, no. 2, pp. 115–148, 1995.

[26] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, 2005.

[27] 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.

[28] 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.

[29] 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.

[30] ——, “Automated innovization for simultaneous discovery of multiple rules in bi-objective problems,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture*

Notes in Bioinformatics), vol. 6576 LNCS. Springer, Berlin, Heidelberg, 2011, pp. 1–15. [Online]. Available: http://www.iitk.ac.in/kangal

[31] 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.

[32] 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.

[33] 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 (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7997 LNCS. Springer, Berlin, Heidelberg, 2013, pp. 1–18.

[34] A. Gaur and K. Deb, "Adaptive use of innovization principles for a faster convergence of evolutionary multi-objective optimization algorithms," in *GECCO 2016 Companion - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*. New York, New York, USA: Association for Computing Machinery, Inc, jul 2016, pp. 75–76. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2908961.2909019

[35] ——, "Towards an automated Innovization method for handling discrete search spaces," in *2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., sep 2015, pp. 2899–2906.

[36] A. Ghosh, K. Deb, R. C. Averill, and E. Goodman, "Combining user knowledge and online innovization for faster solution to multi-objective design optimization problems," Michigan State University, USA, Tech. Rep. COIN Report Number 2020024, 2020.

[37] Z. Tian, Y. Liu, L. Jiang, W. Zhu, and Y. Ma, "A review on application of composite truss bridges composed of hollow structural section members," pp. 94–108, feb 2019.

[38] M. P. Bendsøe, O. Sigmund, M. P. Bendsøe, and O. Sigmund, "Topology design of truss structures," in *Topology Optimization*. Springer Berlin Heidelberg, 2004, pp. 221–259.

[39] M. Z. Abd Elrehim, M. A. Eid, and M. G. Sayed, "Structural optimization of concrete arch bridges using Genetic Algorithms," *Ain Shams Engineering Journal*, vol. 10, no. 3, pp. 507–516, 2019.

[40] R. Cazacu and L. Grama, "Steel Truss Optimization Using Genetic Algorithms and FEA," *Procedia Technology*, vol. 12, pp. 339–346, 2014.

[41] A. C. Lemonge, J. P. Carvalho, P. H. Hallak, and D. E. Vargas, "Multi-objective truss structural optimization considering natural frequencies of vibration and global stability," *Expert Systems with Applications*, vol. 165, p. 113777, mar 2021.

[42] V. Ho-Huu, D. Duong-Gia, T. Vo-Duy, T. Le-Duc, and T. Nguyen-Thoi, "An efficient combination of multi-objective evolutionary optimization and reliability analysis for reliability-based design optimization of truss structures," *Expert Systems with Applications*, vol. 102, pp. 262–272, jul 2018.

[43] S. Bandyopadhyay and A. Mukherjee, "An algorithm for many-objective optimization with reduced objective computations: A study in differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 400–413, jun 2015.

[44] Bright Optimizer - International Student Competition in Structural Optimization, "Optimization Problem of ISCSO 2019," 2019. [Online]. Available: http://www.brightoptimizer.com/problem-iscso2019/

[45] 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.

[46] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms - A comparative case study," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1498 LNCS. Springer Verlag, 1998, pp. 292–301.

[47] D. A. Van Veldhuizen, "Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations," *IRE Transactions on Education*, vol. 8, pp. 125—-147, 1999.

[48] J. D. Gibbons and S. Chakraborti, "Nonparametric Statistical Inference," in *International Encyclopedia of Statistical Science*. Springer Berlin Heidelberg, 2011, pp. 977–979. [Online]. Available: http://unstats.un.org/unsd/snaaaaa/

[49] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric Statistical Methods*, ser. Wiley Series in Probability and Statistics. Wiley, jul 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119196037

[50] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. USA: John Wiley & Sons, Inc., 2001.

[51] G. P. Sutton and O. Biblarz, *Rocket Propulsion Elements*, 9th ed. Wiley, 2016.

[52] 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. [Online]. Available: https://ieeexplore.ieee.org/document/9185861/