Combining User Knowledge and Online *Innovization* for Faster Solution to Multi-Objective Design Optimization Problems

Abhiroop Ghosh[1[0000−0003−3251−5941]], Kalyanmoy Deb[1[0000−0001−7402−9939]], Ronald Averill[1], and Erik Goodman[1[0000−0002−2419−0692]]

Michigan State University, East Lansing MI 48824, USA
ghoshab1@msu.edu,{kdeb,averillr,goodman}@egr.msu.edu
https://www.coin-lab.org/
COIN Report 2020024

Abstract. Real-world optimization problems often come with additional user knowledge that, when accommodated within an algorithm, may produce a faster convergence to acceptable solutions. Modern population-based optimization algorithms, aided by recent advances in AI and machine learning, can also learn and utilize patterns of variables from past iterations to improve convergence in subsequent iterations – an approach termed *innovization*. In this paper, we discuss ways to combine user-supplied heuristics and machine-learnable patterns and rule sets in developing efficient multi-objective optimization algorithms. Two practical large-scale design problems are chosen to demonstrate the usefulness of integrating human-machine information within a multi-objective optimization in finding similar quality solutions as that obtained by the original algorithm with less computational time.

Keywords: Multi-objective optimization · *innovization* · collaborative optimization · knowledge-driven design optimization.

1 Introduction

Multi-Objective Optimization (MOO) algorithms are widely used for solving design optimization problems. A set of trade-off solutions between multiple conflicting objectives, known as Pareto-Optimal (PO) solutions, are the final outcome. This then requires that the decision-makers perform a post-optimal decision-making task to choose a single preferred solution for implementation.

MOO algorithms, on their own and without any additional problem knowledge, can solve real-world design problems involving various practicalities, such as large-dimensional search and objective spaces, nonlinearities, multi-modalities, isolation of optima, etc., but the time taken to achieve a reasonably good solution set may not make the application practical. On the other hand, in most cases, the engineers or domain experts, have critical insights about certain regularity and relationships among variables and objectives, which can be used as key knowledge (or heuristic) in the optimization process. This may result in improved performance and help arrive at an acceptable solution quickly.

Heuristics have been applied across a range of optimization problems covering diverse fields. In combinatorial optimization problems, like the traveling salesman problem, specific studies exist [11]. An algorithm inspired by the knapsack problem has been used in [7] for solving a pharmaceutical clinical trial planning problem. A meta-heuristic optimization algorithm has been used in [2] to achieve collaboration between an engineer and an architect for designing pre-fabricated wall-floor building systems. A two-step heuristic-based algorithm is presented in [12] for performing network topology design.

Innovization [9] is a process of finding hidden common characteristics among the decision variables of current non-dominated solutions which can be retained as important knowledge derived from the optimization procedure, ready to be applied to hasten the convergence properties of an optimization algorithm. In early studies [3, 9], innovization was presented as a post-optimization step to extract key knowledge for future analysis. Further works [10] demonstrated the benefits of applying the derived innovization principles even within the same optimization run in an *online* manner, as heuristics or as repair operators. Here, we combine the two sources of knowledge—human-supplied and machine-derived—to develop an efficient optimization procedure.

The rest of the paper is organized as follows. Section 2 introduces the knowledge-driven design optimization framework. Section 3 introduces our proposed online innnovization approaches. Sections 4 and 5 introduce the truss design and Demand Side Management optimization problems and the results obtained with our proposed approaches. Section 6 presents the conclusions of our study and possible future extensions.

2 Knowledge-driven Design Optimization

The proposed knowledge-driven design optimization process is shown in Figure 1. There are two entities involved in this process: (i) the user and (ii) the optimization algorithm developer. The user can be an engineer or domain expert who describes the problem objectives, constraints, and design parameters (Problem Design Block). The user also often has additional knowledge about good solutions that can be prescribed as additional constraints, but often such information is not entirely quantitative and must be used more as a recommendation, rather than supplied rigid constraints. The algorithm developer is responsible for collecting the problem formulation and any additional problem heuristics from the user (User Knowledge Block) and designing a suitable optimization algorithm (Optimization Process Design Block). Encouraging collaboration between these two entities is the main goal of the Knowledge-driven Design Optimization framework and this paper makes a step towards this goal. Two types of knowledge are used during the optimization: a priori knowledge (U) provided by the user (Past Knowledge Block), and knowledge learned during the optimization (Learning Block). A repair operator (Repair Block) ensures that both types of knowledge are used while generating new solutions (offspring). Both Learning and Repair Blocks constitute the online innovization process. As is evident in Figure 1, any Evolutionary Multi-Objective Optimization (EMO) or Evolution-

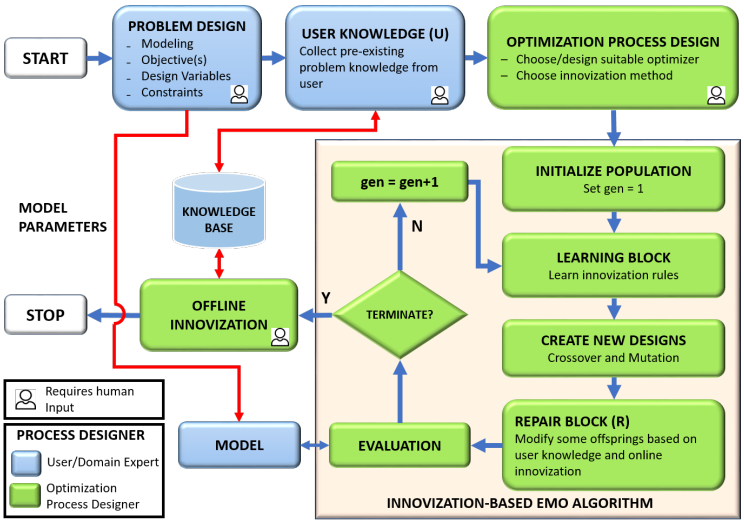


Fig. 1: Knowledge-driven design optimization procedure.

ary Many-Objective Optimization (EMaO) algorithm can be used along with a pattern learning method.

3 Innovization-based Design Optimization Procedure

In a population-based optimization algorithm, non-dominated (ND) solutions represent best solutions of the current population. Hence, they are expected to share some common properties related to the design variables that determine them. For example, coordinates of adjacent nodes in a truss may follow some monotonic property or cross-sectional sizes may follow some symmetric relationships.If such properties are extracted from ND solutions, they can be considered as good derived heuristics, which may or may not have been known and supplied by the designer. If an EMO algorithm makes a correct convergence towards Pareto-optimal solutions, such properties of ND solutions should also firm up and stabilize with generations. Thus, extracting common properties from ND solutions and using them to "repair" future offspring solutions seems to be a useful strategy. Since such a task does not use any additional knowledge from the user, we associate this approach with user knowledge 'None'. With no prior knowledge supplied by the user, the task of identifying such common properties, especially in a space of many design variables, demands a large amount of data, in order to be able to extract a reliable relationship by any machine learning method. However, if the designer provides some indication of likely relationships (such as, symmetry or monotonicity) among a cluster of variables, the learning algorithms are likely to work better. Such a collaboration between a designer with expert knowledge and a machine's ability to extract details should turn out to be a winning strategy, deserving of significant attention.

There are various ways a priori knowledge can be provided to an optimization algorithm. Here, we first discuss the simplest form of providing such knowledge. Initially, the user puts variables into several clusters and for each of them (say, the k-th one C_k), the variables ($\mathbf{x}_k = \{x_i\}$, for $i \in C_k$) within the cluster are expected to possess some unknown relationships for the overall solution to be good or near-optimal. Such unspecified information can also be exploited by a learning algorithm. But if more specific ordering information $I_k(\mathbf{x}_k)$ of variable values $x_i \in C_k$ are available, such as $x_i \leq x_j$, they can also be supplied. More specific relationships, such as $x_i = c_i$, $x_i = 2x_j$ or $\phi(x_i, x_j) = c_{ij}$ (where ϕ and c denote a function and a constant respectively), can also be supplied, if available.

An algorithm can utilize the user-supplied knowledge to any desired extent, from 'lightly' to 'extensively', within an algorithm. An extensive use of supplied knowledge within an optimization algorithm may allow fast convergence, but possibly to a sub-optimal solution, if the supplied knowledge for some reason was not congruent to the true optimal solution set. On the other hand, light use of the supplied knowledge may allow the algorithm to recover from sub-optimal convergence, but may not produce efficient use of the knowledge, if the supplied information was actually correct. In this paper, we propose three repair approaches implementing different extents (low, intermediate, and high) of the use of supplied user knowledge, and investigate their performance.

The proposed repair operations implemented in this study follow a specific procedure. Repair operators are activated once 80% of the population constitutes the non-dominated (ND) set, to prevent any knowledge extraction from low quality solutions. Then, the average ($x_{i_{avg}}$) and standard deviation (σ_i) of the i-th variable values in the current ND set are first computed. If monotonicity or any other specific relationships (e.g., ordering $I_k(\mathbf{x}_k)$ of variable values $x_i \in C_k$) was also supplied, they are noted. Then, at every generation, the offspring population members are repaired using the above information, described next.

3.1 Low-Knowledge Repair Operator

Here we discuss the procedure when no ordering or more specific information is provided. First, we observe whether any pair of variables from the same designated cluster follows the same pattern as they did in the average values derived from the ND set. If variable values of an offspring x_i and x_j ($i, j \in C_k$) have the same relationship as that between $x_{i_{avg}}$ and $x_{j_{avg}}$ of the ND set, no further repair is made. Otherwise, x_i and x_j are repaired as follows:

$$\left(x_{i_{rep}}, x_{j_{rep}}\right) = \begin{cases} \left(\frac{x_i + x_{i_{avg}}}{2}, \frac{x_j + x_{j_{avg}}}{2}\right), & \text{for } (x_{i_{avg}} - x_{j_{avg}})(x_i - x_j) < 0, \\ (x_i, x_j), & \text{otherwise.} \end{cases} \quad (1)$$

The variables x_i and x_j are pushed closer to the average $x_{i_{avg}}$. The same operation is also performed on all other consecutive pairs of variables within the cluster and on other clusters.

Now, we discuss the repair procedure if ordering between x_i and x_j was supplied. If the relationship between x_i^u and x_j^u (for an offspring) and between

$x_{i_{avg}}$ and $x_{j_{avg}}$ (the average of ND solutions) are in agreement, a repair operation will be performed using Eqn. 1; otherwise, Eqn. 2 will be used.

$$\left(x_{i_{rep}}, x_{j_{rep}}\right) = \begin{cases} \left(x_{ij_{avg}} + \frac{x_{ij_{avg}} - x_{i_{avg}}}{2}, x_{ij_{avg}} - \frac{x_{ij_{avg}} - x_{j_{avg}}}{2}\right), & \text{if } x_i^u \geq x_j^u, \\ \left(x_{ij_{avg}} - \frac{x_{ij_{avg}} - x_{i_{avg}}}{2}, x_{ij_{avg}} + \frac{x_{ij_{avg}} - x_{j_{avg}}}{2}\right), & \text{if } x_i^u < x_j^u, \end{cases} \tag{2}$$

where $x_{ij_{avg}} = 0.5\left(x_{i_{avg}} + x_{j_{avg}}\right)$.

3.2 Intermediate-Knowledge Repair Operator

This repair operator follows a similar update of offspring variables, but uses a closer relationship of the variable pairs obtained from the history of the optimization run. In this case, the repair operator for two variables x_i and x_j is similar to that shown in Eqn. 1, but $x_{i_{avg}}$ is replaced by $x_{i_{ref}}$, calculated by adding a momentum parameter (γ) as shown in Eqn. 3.

$$x_{i_{ref}}(t) = x_{i_{avg}}(t) + \gamma v(t), \tag{3}$$

where $v(t) = x_{i_{avg}}(t) - x_{i_{avg}}(t-1)$. If the ordering knowledge between x_i and x_j is available, the same procedure is followed as in the low-knowledge usage operator, but $\mathbf{x}_{avg}$ is replaced with $\mathbf{x}_{ref}$.

3.3 High-Knowledge Repair Operator

This repair operator updates offspring variable vectors more closely to the history of their evolution statistically by staying within one standard deviation (σ_i) of the reference variable value $x_{i_{ref}}$: $[x_{i_{ref}} - \sigma_i, x_{i_{ref}} + \sigma_i]$, where σ_i is the standard deviation of x_i in the ND set:

$$\left(x_{i_{rep}}, x_{j_{rep}}\right) = \left(\mathcal{U}(x_{i_{ref}} - \sigma_i, x_{i_{ref}} + \sigma_i), \mathcal{U}(x_{j_{ref}} - \sigma_j, x_{j_{ref}} + \sigma_j)\right), \tag{4}$$

where $\mathcal{U}(a, b)$ is a uniformly distributed number between a and b. If the ordering knowledge between x_i and x_j is available, we use the same process as in the low-knowledge repair operator, but using $x_{i_{ref}}$, rather than $x_{i_{avg}}$.

We now present the results of an EMO algorithm with the above three repair operators on two engineering design problems.

4 Truss Design Problem

The large-scale truss design problem used in this article is a modified version of the one given in the Bright Optimizer ISCSO 2019 competition [1].
The truss with 260 members and 76 nodes (shown in Figure 2) is simply-supported at two ends and vertical loads of 5 kN are applied to all top nodes in the negative z-direction. The truss length

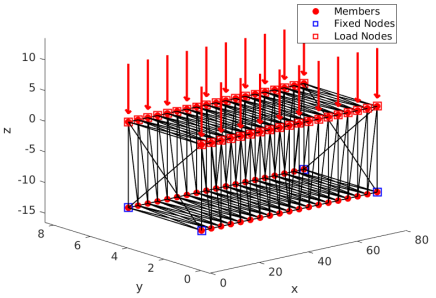


Fig. 2: Truss and its loading.

and width are fixed at 72 metres and 4 metres respectively. The length of the vertical members can vary from 0.5 to 30 metres. A good design is determined by two criteria: low weight and low compliance. There are two sets of design variables: member sizes ($\mathbf{A}$), and coordinates of bottom nodes ($\mathbf{z}$). Finite element analysis (FEA) is performed to compute the weight (f_1), displacement vector ($\mathbf{U}$) and compliance (f_2). The optimization problem formulation is given below:

$$\text{Minimize} \quad (f_1(\mathbf{A},\mathbf{z}),\ f_2(\mathbf{A},\mathbf{z})) = \left(\rho \sum_{k=1}^{N_m} A_k \ell_k(\mathbf{z}),\ \sum_{k=1}^{N_n} \mathbf{F}_k \cdot \mathbf{U}_k(\mathbf{A},\mathbf{z})\right), \tag{5}$$

$$\text{Subject to} \quad \sigma_{max}(\mathbf{A},\mathbf{z}) \leq 248.2 \text{ MPa}, \tag{6}$$

$$|U_k(\mathbf{A},\mathbf{z})| \leq 25 \text{ mm}, \quad \text{for } k = 1,\ldots,N_n, \tag{7}$$

where ρ is the material density, $\ell_k(\mathbf{z})$ is the length of the k^{th} member, which depends on the coordinates of the two end nodes connecting the member, N_m is the total number of members, N_n is the total number of nodes, F_k is the force developed at each node, and σ_{max} is the maximum stress developed in the truss.

4.1 User Knowledge

For the truss shown in Figure 2, any experienced structural engineer, when asked to provide any expert knowledge on properties of good solutions, will think of a few:

1. Symmetry of the truss around a vertical plane at the mid-point in the x-direction and around another vertical plane at the mid-point in the y-direction can be assumed, due to the symmetry of support and loading conditions.
2. Vertical members on the left half should follow a monotonic property to have a smoothly changing shape from the support towards the middle, due to a smooth transfer of load from the middle of the truss toward the end supports.

To inject the above a priori user knowledge into the optimization process, we consider four scenarios as supplied heuristics U_{t1} to U_{t4}. In all scenarios, symmetry of members on left and right of the y-z plane at $x = 36$ (at the mid-point along the x-axis) is provided. This makes the cross-section size of a member and coordinate of a node to the right of the symmetry plane be identical with a symmetrically located member and node to the left of the symmetry plane. The same is also considered for the second symmetry plane mentioned above. This reduces the number of size variables ($\mathbf{A}$) from the original 260 to 86 and the number of node coordinate variables ($\mathbf{z}$) from the original 38 to 10. Thus, the total number of design variables in the modified problem is 96.

In scenario U_{t1}, all 10 (left and front of symmetry planes) z-coordinates of the bottom nodes are grouped in a cluster ($C_1 = \{i\}$ for $i \in [1, 10]$) to indicate that some properties of these z-coordinate values are expected to appear on the Pareto-optimal solutions.

In scenario U_{t2}, further knowledge is provided for the C_1 variable group by stating that the 10 z-coordinates of the bottom nodes are likely to exhibit a monotonic property: the vertical members will monotonically increase in length from the support toward the middle of the truss. Thus, if x_1 is the bottom z-coordinate of the first vertical member and x_2 is the same for the second vertical member and so on, then $x_1 \leq x_2 \leq \ldots$ is additionally assumed.

In scenario U_{t3}, further knowledge can be simulated by providing additional clustering of variables in three more clusters (see Table 1). Scenario U_{t4} uses

Table 1: Knowledge scenarios U_{t3} (with no ordering) and U_{t4} (with ordering).

Cluster	Variable Type	Variable Indices	Ordering (U_{t4})
C_1	z_i of bottom nodes	$[1-10]$	Increasing
C_2	A_i of top Longitudinal members	$[11-19]$	Increasing
C_3	A_i of bottom Longitudinal members	$[20-28]$	Increasing
C_4	A_i of vertical members	$[29-38]$	Increasing

U_{t3} but provides ordering of variables in each cluster as mentioned the final column of the table. The cluster C_1 of variables with a monotonic increase in z-coordinates as in U_{t3} is imposed. Additionally, clusters C_2 and C_3 contain the cross-section size variables of the 10 top longitudinal members (along x) and 10 bottom longitudinal members (along x), respectively, which are assumed to increase monotonically from the support toward the middle. Finally, cluster C_3 contains the cross-section size variable of 10 vertical members, and a similar monotonic increase in their values is assumed. The reason for such an increase in cross-section may come from the fact that the bending moment induced in a simply-supported beam increases from the support toward the middle of the beam and hence a monotonically larger cross-sectional size is needed to withstand a larger bending moment.

4.2 Results and Discussion

NSGA-II [8] is used as the optimization algorithm. Population size is set to 200 and a maximum $2,500$ generations are used with a maximum number of function evaluations (FEs) of 500k. The hypervolume (HV) metric is used to measure the performance. The mean final HV over 10 runs obtained by NSGA-II with no knowledge usage is set as the target performance ($\text{HV}^T = 0.98$). For the HV calculation, both objectives are normalized and the reference point is set as $[1.1, 1.1]$. An algorithm is terminated if the specified HV^T is reached or the maximum allowed FEs of 500k have elapsed.

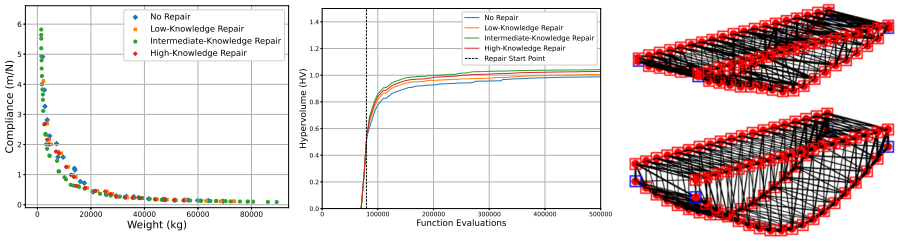
Comparison of the optimization performances is shown in Table 2. Each entry represents the mean and standard deviation of the number of function evaluations taken to reach HV^T. For cases where the algorithm failed to achieve HV^T within 500k FEs, the final HV obtained is additionally mentioned within braces. Each row represents a particular scenario of user-provided knowledge and each column denotes the specific repair operators (described in Section 3) used. In the first row, all columns except the first are marked with 'N/A' since the repair operators cannot operate without any initial user knowledge. From the table, it

Table 2: FEs required to reach target $HV^T = 0.98$.

User Knowledge	Repair operator used			
	None	Low-knowledge	Intermediate-knowledge	High-knowledge
None	500k	N/A	N/A	N/A
U_{t1}	500k	433k ± 17k	**323k** ± 6k	328k ± 2k
U_{t2}	500k	400k ± 8k	**302k** ± 12k	302k ± 12k
U_{t3}	500k	350k ± 10k	**167k** ± 16k	230k ± 12k
U_{t4}	**500k**	500k (0.73)	500k (0.74)	500k (0.74)

can be inferred that, for this problem, for every scenario of user knowledge integration, the intermediate-knowledge operator is able to achieve the target HV with the fewest evaluations. This establishes the fact that an extensive follow-up of the average variable profile (in high-knowledge repair) reduces the diversity of the variable vectors among ND solutions in an evolving population. Also, a low-knowledge-based repair does not exploit the supplied information well. Keeping the intermediate-knowledge repair strategy within NSGA-II, the table also shows that scenario U_{t3} provides the right amount of prior information for NSGA-II with the innovized repair operator to work the best. U_{t4} provides the maximum a priori knowledge to the optimization. However, none of the repair operators was able to achieve HV^T. This shows that too much a priori knowledge constrains the optimization and lowers the variable vector diversity among ND solutions in an evolving population. As in the case of knowledge usage for the repair operators, there exists an optimal level of user-provided knowledge beyond which performance starts to degrade.

Figure 3a shows the ND solutions obtained with U_{t3}. It shows that the intermediate-knowledge operator is able to provide the best quality solutions, dominating those given by the other scenarios. Figure 3b shows the HV evolution with the FEs for U_{t3} with the repair operations being started after 80k FEs. Figure 3c shows the minimum weight and minimum compliance trusses for the intermediate-knowledge repair operator with user knowledge U_{t3}.



(a) ND fronts. (b) HV evolution. (c) Extreme trusses.

Fig. 3: Performance Comparison for User Knowledge U_{t3}. In (c), top is the minimum weight truss and bottom is the minimum compliance truss.

5 Demand Side Management (DSM) Problem

Rise in electric power demand in metropolitan areas implies that existing supply capabilities may struggle to handle peak demands [5]. A battery storage-based DSM method [14] can be a viable technology. In this paper, a simplified multiobjective DSM optimization problem has been created from existing literature [13–15], which will serve as a test problem to demonstrate the benefit of knowledge-driven optimization approaches. An artificial hourly electricity demand (in kWh) dataset over a period of two weeks has been used, as shown in Figure 4b. The daily peak loads are marked in red. A custom version of a real-world tariff model [6] has been used. A battery model based on the one given in [14] has been used.

5.1 Optimization Problem Formulation

A multiobjective optimization problem has been formulated which aims to optimize two conflicting objectives: (i) electricity tariff (C_t), and (ii) battery degradation (L_{deg}).

$$\text{Minimize} \quad C_t(\boldsymbol{P_b}, \boldsymbol{E_d}) = C_e(\boldsymbol{P_b}, \boldsymbol{E_d}) + C_p(\boldsymbol{P_b}, \boldsymbol{E_d}), \tag{8}$$

$$\text{Minimize} \quad L_{deg} = \sum_t \max\{0, L(D(t+1)) - L(D(t))\}, \tag{9}$$

$$\text{subject to} \quad P_{min} <= P_b(t) <= P_{max},\ D_{min} <= D(t) <= D_{max}, \tag{10}$$

$$D(t+1) = D(t) - \eta \times \frac{P_b(t)}{S_b}, \tag{11}$$

where C_t is the total electricity bill, C_e is the energy delivery charges, C_p is the daily peak demand charges, L_{deg} is the total battery degradation, L is the battery degradation loss function, η is the battery charge/discharge efficiency, S_b is the total battery size (kWh), and $E_d(t)$ is the energy demand at time t assumed to be known beforehand. Depth of discharge (D) is defined as the proportion of total battery capacity S_b that has been consumed so far. After a certain number of charge-discharge cycles, the battery capacity degrades to a certain percentage of its initial capacity which makes it unusable. A loss function given in [14] determines the battery degradation for a depth of discharge transition $D(t)$ to $D(t+1)$. $P_b(t)$ is the power supplied by the battery at time t and constitute the decision variable set.

5.2 User Knowledge

In a DSM problem, an intuitive optimal battery operation is to discharge the battery during the peak daily loads. Since the peak demand charges are high [4, 6], the maximum savings can be made with minimal battery degradation if we operate the battery during the peaks. This knowledge (U_{d1}) can be expressed as shown below: $P_b(t_{peak}(i)) = \max_{t\in[24(i-1)+1,\ldots,24i]} P_b(t)$, where $t_{peak}(i) = \arg\max_{t\in[24(i-1)+1,24i]} E_d(t)$ for the i^{th} day. As a part of U_{d1}, multiple clusters are also specified, with each cluster consisting of 24 variables determining the hourly power outputs. For example, if we are performing the optimization over the load data of 1 week, we will have 7 clusters C_1 to C_7 with each cluster $C_i = \{k\}$ for $i \in [1, 7]$ and $k \in [24(i-1)+1, 24i]$.

Scenario U_{d2} is an extension of U_{d1} with the additional knowledge that the battery is charged during the period of lowest demand. This knowledge (U_{d2}) can be expressed as: U_{d1} and $P_b(t_{low}) = \min_{t\in[1,24]} P_b(t)$, where $t_{low} = \arg\min_{t\in[1,\ldots,24]} E_d(t)$. Since there are max and min terms in the above scenario equations, they can also be decomposed into a set of inequality relations as shown below:

$$P_b(t_{peak}) > P_b(t \neq t_{peak}), \quad \forall t \in [1, \ldots, 24], \tag{12}$$

$$P_b(t_{low}) < P_b(t \neq t_{low}), \quad \forall t \in [1, \ldots, 24]. \tag{13}$$

In order to illustrate the effects of adding more information to the optimization, we create a new scenario U_{d3}. This is an extension of U_{d2} where variable ordering relations are imposed between the individual daily peak loads. For every week, during the daily peak demand hours, the power output can be set to be proportional to the demand. For example, if, for 3 days, the peak loads $E_d(t_{peak}(i))$ for $i \in [1, 3]$ in descending order are $\{E_p(3), E_p(1), E_p(2)\}$, the relationship enforced is expressed as $P_b(t_{peak}(3)) \geq P_b(t_{peak}(1)) \geq P_b(t_{peak}(2))$.

5.3 Results and Discussion

This section presents the experimental results on the DSM problem for a two-week period (thus having $24 \times 14 = 336$ variables). NSGA-II [8] has been used as the optimization algorithm to solve this problem. For comparing the performances with different types of user knowledge and repair operators, 10 runs were performed with different random seeds. Maximum FEs was set to be 500k. Identical NSGA-II parameters are used as in the truss problem. HV has been used to measure the performance, and the mean final HV obtained by NSGA-II with no user knowledge is set as the target performance. We set $\text{HV}^T = 0.60$. Performance comparison in terms of FEs required to reach HV^T is made in Table 3.

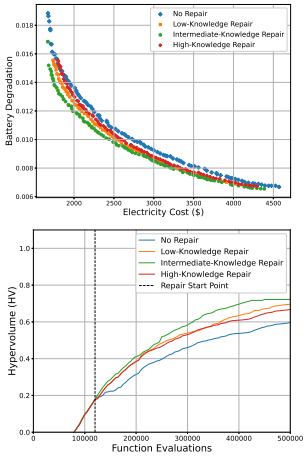
Table 3: FEs required to reach base NSGA-II performance (336-variable DSM problem, $HV^T = 0.60$).

User Knowledge	Repair operator used			
	None	Low-knowledge	Intermediate-knowledge	High-knowledge
None	500k	N/A	N/A	N/A
U_{d1}	500k	412k ± 8.0k	**393k** ± 9.0k	406k ± 6.0k
U_{d2}	500k	366k ± 7.5k	**310k** ± 6.0k	382k ± 12.0k
U_{d3}	**500k**	500k (0.52)	500k (0.51)	500k (0.51)

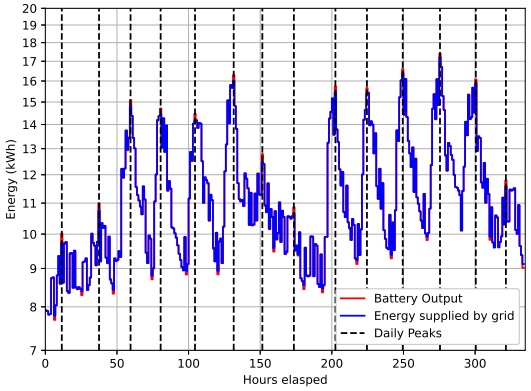
From the table, it is seen that the intermediate-knowledge repair operator is able to reach the target HV in the least number of function evaluations. This supports the conclusions reached for the truss problem that following the average values of the clustered variables too closely constrains the optimization and reduces the diversity of solutions among ND solutions in an evolving population. On the other hand, a low level of knowledge usage does not exploit the available knowledge well enough. Scenario U_{d2} provides the right amount of prior information for the best optimization performance for this problem. U_{d3} provides

the most information, but none of the repair operators were able to achieve the target HV. This shows that incorporating more information than necessary can affect the optimization negatively.

Figs. 4a show the Pareto-front and HV evolution for U_{d2} with the repair operations being started after 120k FEs. It is seen that the intermediate-knowledge operator is able to generate better quality solutions compared to the other algorithms. The intermediate-knowledge operator is able to achieve a higher HV, faster, which is an indicator of its superior performance. Figure 4b shows the energy supplied by the grid and battery for one of the Pareto-optimal solutions. The figure shows that battery output is utilized, as desired by the user, during most of the peak demands.



(a) ND front and HV evolution.



(b) Energy supplied by the grid and battery.

Fig. 4: Performance comparison for user knowledge U_{d2}.

6 Conclusions and Future Work

The study has aimed to demonstrate the usefulness of incorporating user knowledge into an optimization algorithm and has studied the effect of different extents of knowledge usage within an optimization process. Three online innovization-based repair operators have been proposed to exploit any patterns found during the optimization. The proposed methods have been tested on two large-scale practical optimization problems: a 260-member truss optimization problem with 96 variables, and a battery-based DSM problem with 168 variables. The results show the existence of an optimum level of knowledge usage for best performance. If too much or too little knowledge is used, performance is observed to drop. Although an adequate extent of knowledge usage is problem dependent, this study has clearly shown the importance of using the right amount of problem knowledge for a computationally fast optimization process.

Our future work will focus on integrating interactive optimization into the framework such that the user can provide useful feedback during the optimization process. This will further enhance the human-machine collaboration in the

knowledge-driven design optimization process. The three separate repair operators can also be unified under an ensemble method which will remove the need to perform separate experiments. Studies on the generalizability of the proposed methods to other types of practical design problems need to be performed.

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. Bright Optimizer - Ninth International Student Competition in Structural Optimization (ISCSO 2019), http://www.brightoptimizer.com/
2. Baghdadi, A., Heristchian, M., Kloft, H.: Design of prefabricated wall-floor building systems using meta-heuristic optimization algorithms. Automation in Construction **114**, 103156 (2020)
3. Bandaru, S., Deb, K.: Higher and lower-level knowledge discovery from Pareto-optimal sets. In: Journal of Global Optimization. vol. 57, pp. 281–298. Springer (2013)
4. Burke, K.: Consolidated Edison Company of New York, Service Classification No. 9 (2008)
5. Burke, K.: Consolidated Edison Company of New York Inc. Annual Report (2009)
6. Burke, K.: Schedule for Electricity Service General Rules 1 to 23 (2012)
7. Christian, B., Cremaschi, S.: Planning pharmaceutical clinical trials under outcome uncertainty. In: Computer Aided Chemical Engineering, vol. 41, pp. 517–550. Elsevier B.V. (2018)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
9. Deb, K., Srinivasan, A.: Innovization: Innovating design principles through optimization. In: GECCO 2006 - Genetic and Evolutionary Computation Conference. vol. 2, pp. 1629–1636 (2006)
10. Gaur, A., Deb, K.: Effect of size and order of variables in rules for multi-objective repair-based innovization procedure. In: 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings. pp. 2177–2184. Institute of Electrical and Electronics Engineers Inc. (2017)
11. Jaszkiewicz, A., Zielniewicz, P.: Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. European Journal of Operational Research **193**(3), 885–890 (2009)
12. Li, H.: Network topology design. In: Communications for Control in Cyber Physical Systems, pp. 149–180. Elsevier (2016)
13. Tooryan, F., Ghosh, A., Wang, Y., Srivastava, S., Arvanitis, E., Angelis, V.D.: Microgrid Energy Storage Design for Reliability and Cost Performances. In: Proceedings of the 2020 IEEE PES General Meeting (2020)
14. Wang, Y., Song, Z., De Angelis, V., Srivastava, S.: Battery life-cycle optimization and runtime control for commercial buildings Demand side management: A New York City case study. Energy **165**, 782–791 (2018)

15. Yadav, G.G., Gallaway, J.W., Turney, D.E., Nyce, M., Huang, J., Wei, X., Banerjee, S.: Regenerable Cu-intercalated MnO_2 layered cathode for highly cyclable energy dense batteries. Nature Communications **8**(1), 1–9 (2017)