AN INTERACTIVE KNOWLEDGE-DRIVEN MULTI-OBJECTIVE OPTIMIZATION FRAMEWORK FOR ACHIEVING FASTER CONVERGENCE

By

Abhiroop Ghosh

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering – Doctor of Philosophy

2022

ABSTRACT

Users interested in solving real-world optimization problems often have many years of experience. Their intuition or ‘knowledge’ is often overlooked in academic studies due to concerns regarding loss of generality. Such knowledge can be expressed as inter-variable relationships or functions, which can provide some initial guidance to a suitably-designed optimization algorithm. Alternatively, knowledge about variable interactions can also be extracted algorithmically during the optimization by analyzing the better solutions progressively found over iterations – a process termed *innovization*. Any common pattern extracted from good solutions discovered during an optimization run can be used as a repair operator to modify candidate solutions, but the key aspect is to strike a balance between the relevance of the pattern identified and the extent of its use in the repair operator, lest the learned patterns turn out to be properties of unpromising search directions or ‘blind alleys’.

In this dissertation, we propose a framework combining both user-supplied and algorithmically-extracted knowledge to repair solutions during an optimization run in an online fashion. Such a framework is also interactive, allowing the user to provide inputs at any point during the optimization. We show the step-wise modifications required for an evolutionary multi-objective (EMO) framework to allow for: (a) initial user-provided knowledge, (b) automated knowledge extraction and application using innovization methods, and (c) allowing the user to interact with the framework at any point during the optimization run. The path to creating such a framework is systematically performed one step at a time, starting from creating an efficient method of representing problem knowledge, designing a suitable automated innovization procedure, and finally interleaving human-provided and machine-extracted knowledge. We show that such a framework can achieve faster convergence across a variety of practical optimization problems. Some future research directions are also discussed.

This dissertation is dedicated to my parents, friends, and all the other people who directly or indirectly supported my journey.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Kalyanmoy Deb for his invaluable guidance throughout my years as a doctoral student, and for believing in me. I am thankful to my Ph.D. committee members, Dr. Erik Goodman, Dr. Ronald Averill, and Dr. Alejandro Diaz, for their valuable guidance and input on my research. I would also like to thank Ming Liu, Qiren Gao, and fellow COIN lab members for giving me a helping hand when I got stuck. I would also like to acknowledge the financial support of DARPA and the Koenig Chair funding.

I wish to thank my family, without whom I would not be able to reach this far. My teachers from high school and all the way to graduate school deserve a special shout out for their valuable lessons and inculcating the love of learning. I would like to thank my friends in East Lansing, who made this five-year journey special. Many of you have graduated and moved elsewhere, but I will always remember the obligatory Friday hangout and/or the Saturday brunch.

I would also like to acknowledge my friends in high school and in my undergraduate university. Even if we are far away in different countries, the good times with you will never be forgotten.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

DE Differential evolution

DM Decision maker

EA Evolutionary algorithm

EMO Evolutionary multi-objective optimization

GA Genetic algorithm

MOEA Multi-objective evolutionary algorithm

MOO Multi-objective optimization

MOP Multi-objective optimization problem

NSGA Non-dominated sorting genetic algorithm

OPF Optimal power flow

PF Pareto front

PO Pareto optimal

PS Pareto set

SB Stepped beam

SBX Simulated binary crossover

SIV Semi-independent variable

CHAPTER 1
INTRODUCTION

Algorithms required to solve practical optimization problems often need to be customized in order to handle problem-specific difficulties. This is often at odds with academic optimization research, which focuses on finding the best possible solution using generalizable algorithms. While such optimization algorithms can perform reasonably well on practical problems without any modifications [1, 2], their potential is often underutilized by ignoring problem-specific information to avoid the risk of misguiding the algorithm towards sub-optimal search regions. Developing a framework which overcomes this limitation can be very useful, and this dissertation aims to achieve the same.

1.1 Motivation

For practical multi-objective optimization problems (MOPs), existing broad and generic qualitative user knowledge is often ignored while 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 such information with an appropriate quantification during the optimization process, since computational resources may be limited by time and/or cost. Additionally, not using this information may be considered 'insulting' to the user's many years of earned knowledge and intuition about the problem. If the users are ready to utilize additional qualitative user-guided knowledge to speed up the optimization process, challenges exist in determining how to quantify the knowledge and the process by which such knowledge can be incorporated within an optimization algorithm. This study addresses these issues of accepting qualitative user-knowledge and validating their quantification through careful analysis of evolved solutions and also discusses ways to utilize validated user-knowledge to improve solution quality.

It has been discussed in the context of single-objective evolutionary algorithms (EAs) that an optimization algorithm with generic recombination and mutation operators (such as simulated binary crossover (SBX) [3] or differential evolution (DE) [4] may be too slow to lead to high-performing regions of the search space in complex practical problems involving a large number of variables. Although these methods are probably ideal for solving benchmark problems due to non-availability of problem-specific information, real-world problems usually come with physical parameters that must be related in certain ways for a solution to be meaningful. In a recent billion-variable resource allocation problem originating from a casting scheduling task in a foundry [5], the linearity of constraint structures allowed the development of an efficient customized evolutionary algorithm that exploited the linearity. The outcome demonstrated a polynomial performance of the resulting customized genetic algorithm that scaled to solve similar scheduling problems having 50,000 to one billion variables.

Another study used the concept of semi-independent variables [6], 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 satisfies all the above monotonic 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.

Pre-specifying problem information is not the only way to provide guidance to an optimization algorithm. Recent *innovization* studies [7, 8, 9] have demonstrated that additional problem information can be extracted from high-performing solutions created during the optimization process, thereby leading to a more efficient search and faster convergence.

It cannot be denied that users with many years of experience in solving a given problem class can usually provide certain qualitative information about variables, such as monotonicity or other properties among variables. Although qualitative, it will not be wise to simply ignore such information in an optimization task, particularly when the problem being solved is large-scale or sufficiently nonlinear. However, integration of additional qualitative problem information raises several challenging issues. How can the qualitative information be quantified? How can the qualitative information be validated to be useful in speeding progress toward the optimal region? To what extent should such information be used in an optimization algorithm in order not to cause a premature or false convergence? Thus, an adaptive algorithm must assess the worth of user-provided information systematically and utilize it differently as the algorithm navigates from its initial generation to the end.

This study addresses the issues mentioned above and proposes generic knowledge-based methodologies for solving practical multi-objective optimization problems. The aim is to demonstrate how a multi-objective optimization algorithm can work in tandem with pre-specified qualitative problem information, automatically extract problem-specific knowledge, and keep the user in the loop so that large-scale problems can be solved relatively quickly.

1.2 Challenges

The objective of this dissertation comes with a lot of challenges which needs to be addressed in order to apply the proposed methods for practical problems.

1.2.1 Knowledge representation

Collecting any relevant problem knowledge from the user is not useful unless there exists a method to convert qualitative knowledge into a quantitative version. A standardized representation method will allow the user to communicate with the optimization algorithm which can only process mathematical representations of knowledge.

1.2.2 Quality of user-provided knowledge

While user guidance in the form of problem knowledge can be useful, it should not hinder the ability of an optimization algorithm to discover new solutions. Too much emphasis on user-provided knowledge is not desirable as it can be imperfect or incorrect.

1.2.3 Knowledge discovery and application

A good automated method to learn useful information from the good solutions found during the intermediate runs can be very useful towards speeding up convergence. However, the same concerns as those regarding user-provided knowledge apply in this case. Any discovered knowledge should not be trusted blindly since it can potentially steer the algorithm towards subpar solutions. Applying knowledge learned from solutions obtained prior to convergence should be done carefully.

1.2.4 User interaction method

In order to allow for effective user interaction, we need to develop methods to allow seamless information transfer between optimization algorithms and the user. Some safeguards need to be present to verify the user-provided information is of high quality.

1.2.5 Large-scale problems

For problems with a high number of decision variables, interaction with the user can become complex. The computational cost of knowledge discovery also increases, thus requiring scalable solutions.

1.3 Organization of the dissertation

The rest of dissertation is structured as follows. In Chapter 2, we provide some concepts and literature survey relevant to the dissertation topic. In Chapter 3, we define multiple ways to represent problem knowledge in order to standardize our experiments. In Chapter 4, we present the first iteration of our proposed framework which provides a way for the user to provide some initial guidance to the optimization algorithm, as well as propose an online innovization method to learn certain types of inter-variable relations and apply them through one or more repair operators. Chapter 5 consists of the final version of the proposed framework which is able to handle multiple types of knowledge representation. Chapter 6 shows a software implementation of the proposed framework. Finally we summarize the contributions of this dissertation and provide some promising directions for future works in Chapter 7.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

In this chapter, we outline the necessary concepts and related works.

2.1 Multi-objective optimization

For real-world optimization problems, there are often multiple, potentially conflicting criteria or objectives a good design needs to satisfy [10, 11]. Such problems are also referred to as multi-objective optimization (MOO) problems. In traditional single-objective optimization algorithms, only one objective function can be optimized which poses certain limitations for solving MOO problems. Approaches like weighted metric methods [12] can be used to tackle such problems using single-objective optimization algorithms. On the other hand, true multi-objective optimization algorithms like MOGA [13], SPEA [14] and NSGA-II [15] aim to present a set of solutions to the decision maker (DM) representing a tradeoff between two or more objectives. Out of these, one of the solutions will be selected by the DM. An MOO problem can be formulated as follows:

$$\text{Minimize } f_m(\mathbf{x}) \qquad m = 1, 2, ..., M, \tag{2.1}$$

$$\text{s.t. } g_j(\mathbf{x}) \geq 0, \qquad j = 1, 2, ..., J, \tag{2.2}$$

$$h_k(\mathbf{x}) = 0, \qquad k = 1, 2, ..., K, \tag{2.3}$$

$$x_i^L \leq x_i \leq x_i^U, \quad i = 1, 2, ..., N, \tag{2.4}$$

where $\mathbf{x} = [x_1, x_2, ..., x_N]^T$ is a solution represented as a vector of N decision variables, $f_m(\mathbf{x})$ represents an objective function, $g_j(\mathbf{x})$ represents an inequality constraint, $h_k(\mathbf{x})$ represents an equality constraint, M is the number of objectives, J is the number of inequality constraints and K is the number of equality constraints. Equation 2.1 degenerates to a single-objective optimization problem formulation for $M = 1$. Figure 2.1 represents a bi-objective optimization problem ($M = 2$) and the mapping from the feasible decision variable space to the objective space.

In case of a single objective optimization problem, two solutions can be compared using the corresponding objective values. For a minimization problem, a lower objective value is considered to be better. However for the MOO problem formulation shown in Equation 2.1, such comparison

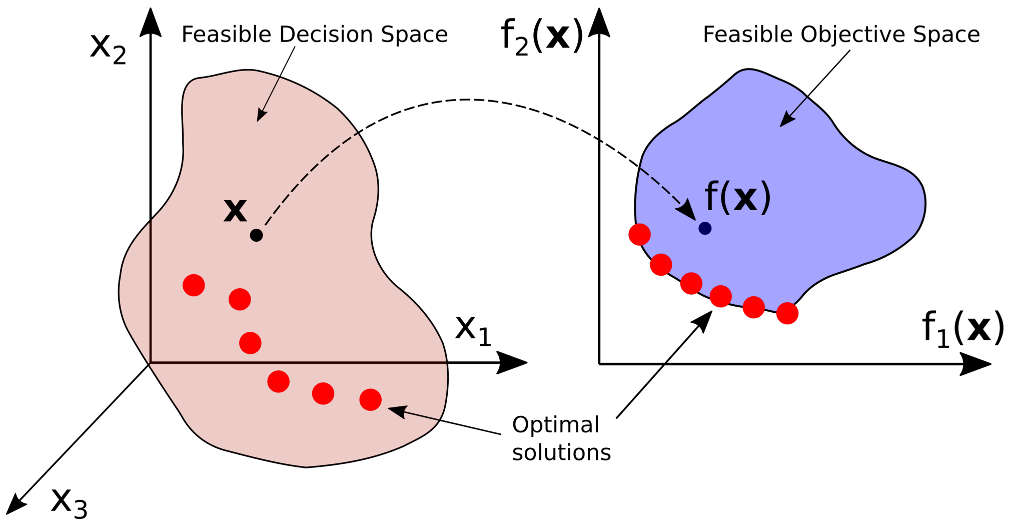


Figure 2.1 Decision and objective space of a multi-objective optimization problem.

is not as straightforward as the single-objective optimization case. For a bi-objective problem ($M = 2$), if one solution ($\mathbf{x}^{(1)}$) has a lower value of all the objectives ($f_1(\mathbf{x})$ and $f_2(\mathbf{x})$) compared to another solution ($\mathbf{x}^{(2)}$), then $\mathbf{x}^{(1)}$ can be declared as superior to $\mathbf{x}^{(2)}$. Alternately, $\mathbf{x}^{(1)}$ may have a lower value for the first objective ($f_1(\mathbf{x})$) compared to $\mathbf{x}^{(2)}$, but a higher objective value for the second objective ($f_2(\mathbf{x})$). In such a case, $\mathbf{x}^{(1)}$ cannot be said to be better than $\mathbf{x}^{(2)}$. To resolve this issue, the concept of dominance [11] can be used as defined below.

Definition 1 *A solution $\mathbf{x}^{(1)}$ is said to dominate another solution $\mathbf{x}^{(2)}$, if $f_i(\mathbf{x}^{(1)}) \leq f_i(\mathbf{x}^{(2)}) \forall i \in \{1, 2, ..., M\}$ and $f_j(\mathbf{x}^{(1)}) < f_j(\mathbf{x}^{(2)}) \exists j \in \{1, 2, ..., M\}$. $\mathbf{x}^{(1)}$ dominating $\mathbf{x}^{(2)}$ is represented as $\mathbf{x}^{(1)} \preceq \mathbf{x}^{(2)}$*

Definition 2 *A set of solutions P is said to be non-dominated if no member of P is dominated by another member of the same set.*

In simple terms, definition 1 says that a solution $\mathbf{x}^{(1)}$ is better than or 'dominates' another solution $\mathbf{x}^{(2)}$ if the following conditions are satisfied:

1. $\mathbf{x}^{(1)}$ is *equal to or better than* $\mathbf{x}^{(2)}$ in all the objectives $[f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})]$.

2. $\mathbf{x}^{(1)}$ is *strictly better* (has a lower value) than $\mathbf{x}^{(2)}$ in at least one objective.

Definition 2 defines the concept of a non-dominated set where no solution in the set is better than the other according to the dominance criteria defined in definition 1.

Figure 2.2 illustrates the dominance concept by showing 6 solutions ($\mathbf{x}^{(1)}$ to $\mathbf{x}^{(6)}$) in the objective space. After performing pairwise comparison among all the solutions, the non-dominated set $P = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}\}$. Each solution in P dominates the other solutions not in P. For example, according to the dominance criteria shown in definition 1, $\mathbf{x}^{(2)} \preceq \mathbf{x}^{(5)}$ and $\mathbf{x}^{(2)} \preceq \mathbf{x}^{(6)}$.

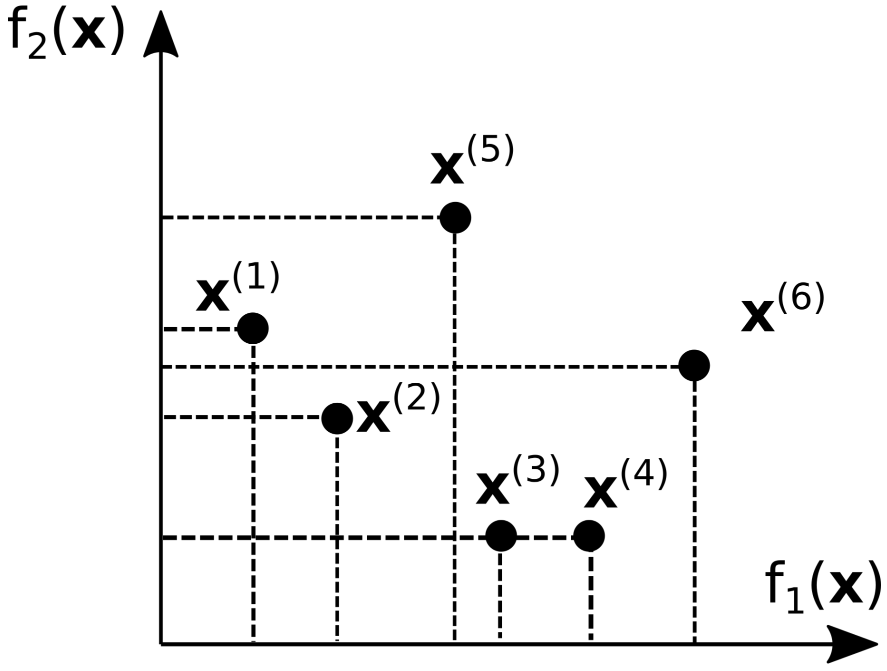


Figure 2.2 A multi-objective optimization problem.

2.2 Evolutionary multi-objective optimization (EMO) algorithms

Practical design problems often contain complexities like multiple local optima, discrete variables, non-linearity, discontinuity, etc. Classical methods like gradient descent [16] can be ill-equipped to handle such situations. An evolutionary algorithm (EA) like the genetic algorithm (GA) [17], genetic programming [18], evolution strategy [19], and differential evolution [4] does not have such limitations. A significant feature of EAs is the use of multiple solutions across iterations or generations as opposed to point-based methods. Such a population of potential solutions allows EAs to explore the search space with a reduced risk of converging to a local optimum.

Many practical problems often have multiple conflicting objectives and can be designed as

MOO problems. Population-based EAs are able to store and evaluate multiple solutions at each generation, a property suitable for solving MOO problems. While methods such as weighted sum and Tchebycheff metrics [12] can be used to apply single-objective EAs to MOO problems, they often have some shortcomings [11]. Thus, evolutionary multi-objective optimization (EMO) algorithms [11, 10, 15, 20, 21, 22, 23] were proposed in order to tackle MOO problems.

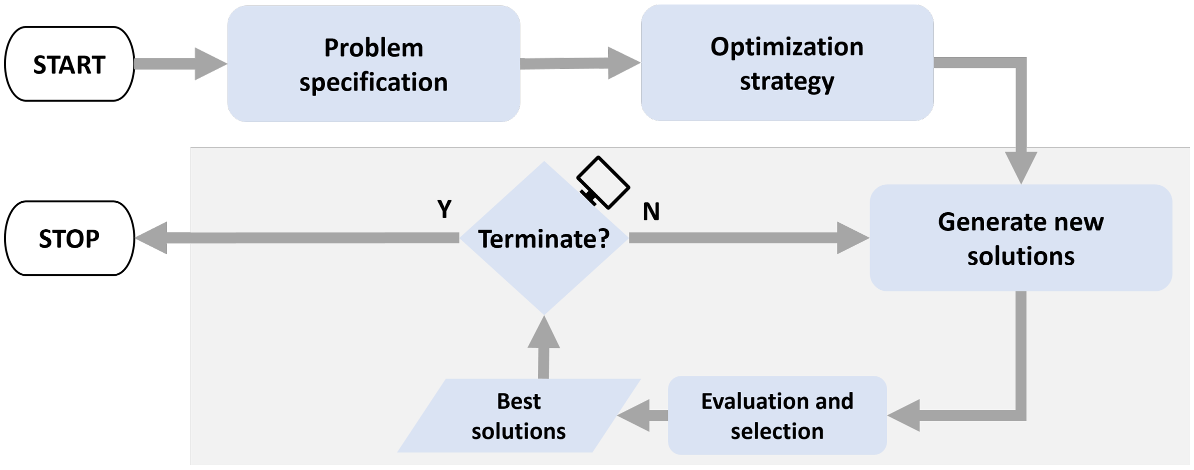


Figure 2.3 A basic EMO framework.

A basic EMO framework is illustrated in Figure 2.3. It consists of multiple components as described below:

- 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.
- Optimization strategy - Here, an EA is selected with suitable parameter values for solving the problem.
- Generate new solutions - Create new candidate solutions according to the EMO algorithm. For genetic algorithm-based EAs, new solutions are generated randomly in the first generation and through crossover and mutation operators in the subsequent generations.
- Evaluation and selection - Evaluate the objective functions and select best-performing solutions.

- Best solutions - These are the best-performing solutions selected in the evaluation and selection step. For an EMO algorithm the non-dominated solution set consists of the best-performing solutions.

- Termination criteria - Decide when to stop the algorithm. In this dissertation, an upper bound on the number of objective function evaluations (FEs) performed by the EMO algorithm will be used as a termination criteria.

2.3 Human-computer interaction in optimization

For real-world problems, designers often have some problem knowledge or intuition developed over multiple years. Developing methods to exploit such knowledge can help in improving the optimization performance. In the following sections, we review existing work which proposes methods to incorporate problem knowledge into an optimization process, either through human input or via heuristics. We also review works which propose automated methods to algorithmically extract problem knowledge.

For complex single-objective practical problems, evolutionary algorithms (EAs) with generic recombination and mutation operators [3, 4] may be too slow to lead to high-performing regions of the search space. Good performance of an algorithm in solving benchmark problems such as ZDT [24], DTLZ [25], WFG [26], and CEC 2009 [27] does not always translate to good performance on practical problems. For such cases, creating customized algorithms leveraging additional problem information is necessary. Deb and Myburgh [28] proposed a customized evolutionary algorithm that exploited the linearity of constraint structures to solve a billion-variable resource allocation problem. A micro-genetic algorithm [29] combining range-adaptation and knowledge-based re-initialization was applied to an airfoil optimization problem. Semi-independent variables [6] can be used to handle user-specified monotonic relationships among variables in the form of $x_i \leq x_{i+1} \leq x_{i+2} \leq \ldots \leq x_j$. Every newly created solution is guaranteed to automatically satisfy the given monotonic constraints. However, a new description of semi-independent variables needs to be defined and implemented for a different relationship, such as a monotonically decreasing relation of the form $x_i \geq x_{i+1} \geq x_{i+2} \geq \ldots \geq x_j$. Some techniques for combining EAs with problem knowledge are given in [30].

2.3.1 Human-specified problem knowledge

A common way to incorporate human intuition into an optimization algorithm for a practical problem is to design a customized EA or apply some heuristics. Customized algorithms have problem knowledge implicitly built-in, and often provide a better performance compared to the standard versions of popular EAs. Amouzgar et al. [31] proposed a modified GA with a customized solution representation for a multi-objective machining tool-indexing problem. Diaz et al. [32] proposed a simulation-based NSGA-II algorithm for optimizing reconfigurable manufacturing systems. Nourmohammadi et al. [33] created a a hybrid GA with variable neighborhood search in order to solve an integrated supermarket location and transport vehicles selection problem. It was shown to outperform traditional GA. Amouzgar et al. [34] proposed a GA with custom operators for minimizing the non-machining time of CNC machines. In combinatorial optimization problems, like the traveling salesman problem, specific studies exist [35]. An algorithm inspired by the knapsack problem has been used in [36] for solving a pharmaceutical clinical trial planning problem. A meta-heuristic optimization algorithm has been used in [37] 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 [38] for performing network topology design. Another application exists in the layer patterning of plate fin heat exchangers [39]. Gandomi et al. [6] proposed a novel semi-independent variable formulation in order to enforce pre-defined monotonically increasing or decreasing variable patterns.

2.3.2 Computer-extracted problem knowledge

In the previous sections, we have covered some works which made use of customized EAs or problem formulations to incorporate additional problem knowledge. In this section, we review some existing works which do not make use of pre-specified problem knowledge. Instead, learning methods are used to automatically extract such information. Rios et al. [40] proposed a novel deep neural network for learning compact geometric representations of car shapes which can be used to generate more optimal designs. Friess et al. [41] used artificial neural networks for feature extraction among optimal solutions found by an EA. A simple transfer learning framework for continuous evolutionary single-objective optimization is proposed in [42] which captures knowledge from completed optimization runs as empirical probability distributions. Ruan et al. [43] performed a computational study to determine the effectiveness of knowledge transfer from one instance of an optimization run to another.

Alternatives to pre-specifying problem information exist, such as cultural algorithms which encode domain knowledge inside a belief space [44]. Self-organizing maps (SOMs) can provide information about important design variable clusters [45]. Recent *innovization* studies [7, 8, 9] aim to extract additional problem information from high-performing solutions during the optimization process. This can lead to a faster convergence.

2.3.3 Interactive optimization

For practical problems, designers may want to play a more active role in the optimization process rather than just providing information in an *apriori* manner. In this section, we review some existing works covering interactive optimization involving the exchange of problem knowledge.

Saha et al. [46] proposed a method to mimic user behavior when using an EA to find the optimal shapes of 3D objects which satisfies the user's aesthetic sensibilities. Karlsson et al. [47] combines NSGA-II, and a customized data mining algorithm, called Flexible Pattern Mining to extract knowledge in the form of rules and guide the optimization to converge towards a decision maker's preferred region in the objective space. Morshedzadeh et al. [48] propose a method to optimize information management in product lifecycle management systems, which is consistently being changed and revised by users. Lidberg et al. [49] proposes a method to support the DM during the optimization real-world factory flows. Smedberg et al. [50] proposes Trend Mining – a visualization method aimed towards assisting DMs in understanding the effect of each variable on the objective space.

Interactive optimization is when the user, referred to as the decision maker (DM), provides guidance during the optimization [51]. Multiple ways to interactively specify information exist, such as aspiration levels [52], importance of individual objectives [53, 54], pairwise comparison of solutions [55], etc. Many other interactive optimization methods can be found in literature [56, 57, 58, 55, 59, 60, 61, 62, 63].

Using additional problem information comes with a set of challenges. An effective knowledge representation method needs to be designed which can be used effectively by an optimization algorithm. At the same time, it should also be comprehensible to the user. Validating any user-provided knowledge is necessary since the quality of supplied information may vary. This reduces the possibility of a premature or false convergence. The user may wish to periodically monitor and review optimization progress as well as any learned information. If necessary, the user can also supply information in a collaborative fashion [58, 64]. However, care needs to be taken to ensure that any user feedback does not lead the search process towards sub-optimal solutions.

2.3.4 Innovization

Non-dominated solutions found by an MOEA may share common characteristics which may help us uncover hidden knowledge about the practical problem under consideration. *Innovization* [7] is the process of finding such hidden common characteristics, also referred to as ‘rules’, which can be retained as important knowledge, ready to be applied to hasten the convergence properties of an optimization algorithm. In early studies [8, 7], innovization was presented as a post-optimization step to extract key knowledge for future analysis. The problem first needed to be solved using an MOEA. Then a manual analysis was performed to discover hidden relationships existing among the decision variables, constraints and objective functions.

Automated innovization was introduced in [65] which allowed the discovery of multiple design rules in a single step. Some applications of automated innovization on practical engineering problems were covered in [66].

Multiple Pareto fronts can be obtained from the same optimization problem when the problem parameters are changed. Higher level innovization [67, 68] is the process of finding common overarching rules across multiple such Pareto fronts. Lower level innovization [8, 69], on the other hand, involves finding hidden characteristics from a preferred region of a single Pareto front. Temporal innovization [70] determines the relative hierarchy of obtained rules by analyzing their significance across previous generations. The principles appearing earlier are hierarchically more important than those which emerge later in the optimization.

Further works [71, 72, 73] demonstrated the benefits of applying the derived innovization principles within the same optimization run in an *online* manner, as heuristics or as repair operators. It was shown that online application of innovization rules helped in achieving faster convergence on many benchmark and practical problems.

2.3.5 Optimization software

Multiple software implementations of optimization algorithms exist in order to make existing single and multi-objective optimization algorithms accessible to academic and commercial users. A well-designed optimization software or 'solver' can often make formulating and solving optimization problems easier, as well as save time. In addition, they can provide useful visualization tools and parallel computation capabilities.

Some of the popular commercial optimization solvers are CPLEX [74], GUROBI [75], XPRESS [76], modeFrontier [77], VisualDoc [78], HEEDS [79], Optimus [80], optiSLang [81], and iSight [82], to name a few. Open-source solvers also exist, such as jMetal [83, 84], pymoo [85], PISA [86], NIMBUS [87], and DESDEO [88].

The popularity of optimization solvers illustrate the importance to back up theoretical optimization frameworks with a robust software implementation. As a part of this dissertation, a graphical user interface (IK-EMOViz) was also developed.

CHAPTER 3

KNOWLEDGE REPRESENTATION

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

3.1 Structure of rules considered in this study

For an interactive knowledge-based optimization algorithm to work, a standard form of knowledge representation is necessary which is simple enough for users to understand but has enough complexity to capture problem knowledge accurately. Using algebraic expressions or 'rules' is one way of representing knowledge and has been extensively used in the 'innovization' literature [7, 9]. A rule can take the form of an equality or an inequality, as shown below:

$$\phi(\mathbf{x}) = 0, \tag{3.1}$$

$$\psi(\mathbf{x}) \leq 0. \tag{3.2}$$

Any arbitrary form of rules involving many variables from a decision variable vector ($\mathbf{x}$) and complicated mathematical structures of functions ϕ or ψ may be considered [89, 90], but such rules would not only be difficult to learn, they would also be difficult to interpret by the user. In this study, we restrict the rules to have simple structures involving a maximum of two variables, as discussed below.

3.1.1 Constant rule

This type of rule involves only one variable taking a constant value ($x_i = \kappa_i$). In terms of Equation 3.1, for the i-th variable, the structure of the rule becomes $\phi_i(\mathbf{x}) = x_i - \kappa_i$. This type of rule can occur if multiple high-performing solutions are expected to have in common a fixed value of a specific variable [91].

3.1.2 Power law rule

Power law rules [7] for two variables x_i and x_j can be represented by Equation 3.1 as $\phi_{ij}(\mathbf{x}) = x_i x_j^b - c$, where b and c are constants. This form makes power laws versatile enough to encode a wide variety of rules, such as proportionate or inversely proportionate relationships among two variables. Interestingly, an inequality power law using a ψ function can also be implemented, but such a rule may represent a relationship loosely and we do not consider it here.

3.1.3 Equality rule

This type of rule can express the equality principle of two variables x_i and x_j observed in high-performing solutions. In terms of Equation 3.1, $\phi_{ij}(\mathbf{x}) = x_i - x_j$ is the rule's structure.

3.1.4 Inequality rule

This type of rule can represent relational properties of two variables x_i and x_j as $x_i \leq x_j$ or $x_i \geq x_j$. In terms of Equation 3.2, $\psi_{ij}(\mathbf{x}) = x_i - x_j$ or $\psi_{ij}(\mathbf{x}) = x_j - x_i$ are the respective rules. For example, the radius of two beams in a truss [92] might be related via this type of rule.

After describing the chosen rule structures, we are now ready to discuss the procedures of extracting such rules from high-performing variable vectors and applying the extracted rules to the optimization algorithm. A summary of their representations and use in our analysis are provided in Table 3.1.

3.2 User-provided knowledge

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

Table 3.1 Rule types and the corresponding mathematical representation. $\mathcal{X}$ represents the set of ND solutions. x_i and x_j refer to the i-th and j-th variables, respectively, of an ND solution $\mathbf{x} \in \mathcal{X}$. The corresponding variables in a new solution $\mathbf{x}_r$ to be repaired are labeled as x_{ir} and x_{jr}, respectively. Normalized variables are represented by a hat $(\hat{x}_i, \hat{x}_j)$. Higher ranked rules are preferred while performing repair. The score (s) is a measure of how well $\mathcal{X}$ follows the rule in the representation column. Satisfaction condition dictates whether $\mathbf{x}_r$ follows the respective rule.

Rule type	Representation	Score
Constant	$\phi_i(\mathbf{x}) = x_i - \kappa_i = 0$	$s_{\phi_i} = \frac{\lvert A_i \rvert}{\lvert \mathcal{X} \rvert}$, where $A_i = \{1 : \forall \mathbf{x} \in \mathcal{X}, \lvert x_i - \kappa_i \rvert \le \rho_i\}, \kappa_i = \tilde{x}_i$
Power law	$\phi_{ij}(\mathbf{x}) = \hat{x}_i \hat{x}_j^b - c = 0$	$s_{\phi_{ij}} = R^2$ score of linear regression shown in Equation 5.2
Equality	$\phi_{ij}(\mathbf{x}) = x_i - x_j = 0$	$s_{\phi_{ij}} = \frac{\lvert B_{ij} \rvert}{\lvert \mathcal{X} \rvert}$, where $B_{ij} = \{1 : \forall \mathbf{x} \in \mathcal{X}, \lvert x_i - x_j \rvert \le \varepsilon_{ij}\}$
Inequality ($\le$)	$\psi_{ij}(\mathbf{x}) = (x_i - x_j) \le 0$	$s_{\psi_{ij}} = \frac{\lvert C_{ij} \rvert}{\lvert \mathcal{X} \rvert}$, where $C_{ij} = \{1 : \forall \mathbf{x} \in \mathcal{X}, x_i \le x_j\}$
Inequality ($\ge$)	$\psi_{ij}(\mathbf{x}) = (x_j - x_i) \le 0$	$s_{\psi_{ij}} = \frac{\lvert D_{ij} \rvert}{\lvert \mathcal{X} \rvert}$, where $D_{ij} = \{1 : \forall \mathbf{x} \in \mathcal{X}, x_i \ge x_j\}$

3.2.1 Variable grouping

For a problem with n variables, there can be $\frac{n(n-1)}{2}$ pairwise variable interactions. For any reasonable-sized problem, such a huge number of meaningful relationships may not exist. In practice, the user may be interested in only a handful of relationships that relate some critical decision variables. In order to reduce the complexity, variables can be divided into different groups G_k for $k = 1, 2, ..., n_g$. Each group consists of variables that the user thinks are likely to be related. Group information is specified prior to the optimization. If no group specification exists, then all N variables are considered as part of a single group and all $\frac{n(n-1)}{2}$ pairwise variable combinations will be considered. Inter-group relationships are not discoverable under this scheme. Variables that are not part of any group are assumed not to be related to other variables. For example, assume there are two variable groups $G_1 = \{2, 3, 5\}$ and $G_2 = \{1, 4, 7\}$ for an 8-variable problem. For G_1 all pairwise combinations (x_2, x_3), (x_2, x_5), and (x_3, x_5) will be checked for the existence of any possible relationships. A similar process is repeated for G_2. Since inter-group relationships are not explored, combinations like (x_3, x_7) will not be considered. Variables x_6, and x_8 are not part of any group, hence they are assumed not to be related to the other variables in any meaningful way.

3.2.2 Rule hierarchy

In the proposed framework we consider four types of rules as presented in Section 3.1. A pair of variables may be related by more than one rule type. In that case, we will select one type of rule according to a predefined hierarchy. At the start of the optimization each rule is assigned a rank. The existence of a particular rule type for one or more variables is checked rank-wise. For example, if constant rules are ranked 1, followed by power laws (rank 2) and inequalities (rank 3), then the variables in every group will be checked for constant rules first. The variables which do not exhibit constant rules will then be checked for power laws, and so on. For relations having equal ranking, a scoring criterion needs to be used to determine which rule better represents the non-dominated (ND) front and will be used by the algorithm.

CHAPTER 4
USER-GUIDED INNOVIZATION-BASED MOEA FRAMEWORK

In this chapter, we propose an extension of the base EMO framework described in Chapter 2.2 which combines initial user-provided guidance and online innovization to perform a more efficient search. It also makes provisions for verifying the correctness of each piece of provided information, adjusting its influence on the optimization process accordingly. This requires an optimization algorithm that is customizable. Classical point-based methods do not offer such flexibility, making MOEAs such as NSGA-II [15] a better choice. Fig. 4.1 illustrates the MOEA/I framework and covers the entirety of the optimization process.

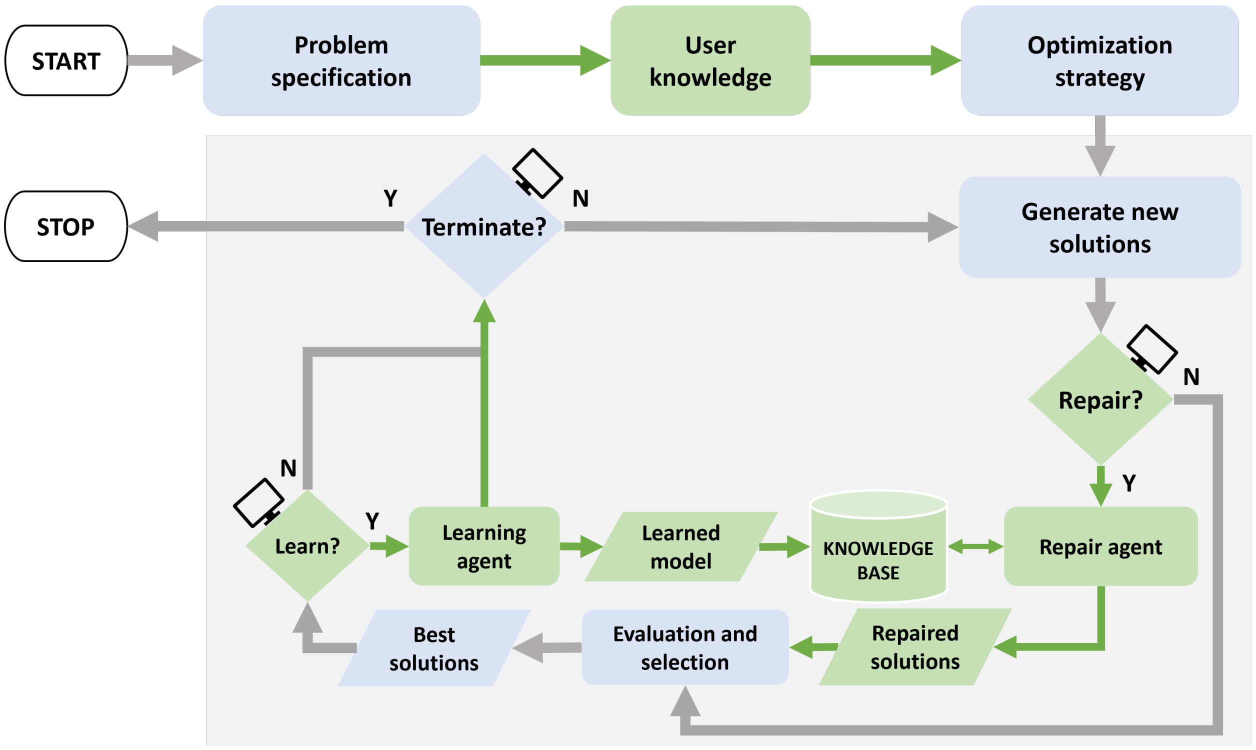


Figure 4.1 User-guided innovization-based MOEA framework (MOEA/I) showing learning and repair agents. Blue blocks represent a normal EMO. Green blocks represent the components responsible for knowledge extraction and application.

The MOEA/I framework [93] consists of some major additions to the base EMO framework as described below:

- User knowledge - An intermediate step between problem specification and devising an optimization strategy where the user can provide some initial guidance to the optimization algorithm.

- Learning agent - An algorithm or a machine learning method that extracts knowledge from the best solutions found during the optimization.
- Learned model - The model created by the learning agent which encodes knowledge in some pre-specified format.
- Knowledge base - Information about the learned models in each generation is stored in the knowledge base. This information can be retrieved later if necessary.
- Repair agent - Based on the information stored in the knowledge base, the repair agent modifies or 'repairs' the decision variable values of the newly generated solutions.
- Repaired solutions - The modified solution set to be sent for evaluation.

The subsequent sections present the user knowledge specification procedure and the four repair operators used in this study, three of which are an extension of the operators proposed by [92]. Each repair operator functions differently when explicit relations are specified compared to when only variable groups are specified without any explicit relation. In this study, the scope of the repair operators is limited to inequality relations. However, if the problem demands it, custom repair operators can be designed and plugged into the MOEA/I framework.

4.1 User knowledge

User knowledge can be of different types, relating two or more *comparable* variables or a combination of variables, objective or constraint functions in certain ways. Comparable variables mean that they are of identical units and varying in similar range. For example, two size-related variables varying within $[a, b]$ are defined as comparable variables here. This information is in addition to the optimization problem formulation provided by the user and perhaps acquired over many years of experience of the user in dealing with past solutions of the problem.

An upper-diagonal relationship matrix $U = [u_{ij}]$, where $i, j \in [1, n], i < j$ is proposed to store all user-provided pair-wise variable information:

$$u_{ij} = \begin{cases} 0, \text{ if } x_i \text{ and } x_j \text{ are not likely to have any relationship,} \\ 1, \text{ if } x_i \text{ and } x_j \text{ have a relationship, but unknown,} \\ 2, \text{ if } x_i < x_j, \\ 3, \text{ if } x_i > x_j, \\ 4, \text{ if } x_i \approx x_j. \end{cases} \tag{4.1}$$

As mentioned before, all variable pairs for which the relationship index $u_{ij} > 0$ are required to have the same scale and represent similar quantities. This is justified by some practical considerations. Two variables representing totally different quantities, such as length and weight, should not be expected to have any direct inequality relationship.

For LSMOPs, a large number of decision variables will make specifying all $\frac{n(n-1)}{2}$ relationships extremely cumbersome. However, in practical problems, variable patterns can be specified or envisioned to have relationships in groups (or variable clusters). K groups of variables can be specified (G_k, $k = 1, 2, \ldots, K$). An example of a 6-variable Umatrix and 3 groups (G_1-G_3) is given in the supplementary document.

For the unknown relationship ($u_{ij} = 1$), it is expected that the optimization task will analyze its best population members to try to discover an exact relationship between variables x_i and x_j. For $u_{ij} = 2$ to 4, the optimization task is expected to establish whether the specified relationship exists and utilize the relationship to fix any population member that violates it. Thus, the qualitative relationships provided at the start of the optimization task must first be validated and utilized to make a faster convergence. As a by-product, the user also gets a quantitative version of the relationships they provided. This repair-based quantification process is described below.

As an example, consider 6 decision variables divided into three groups of variables G_1 =

$\{1,2,3\}$, $G_2 = \{1,5,6\}$ and $G_3 = \{4,6\}$. The matrix U can be defined as:

$$\mathbf{U} = \begin{bmatrix} 4 & 3 & 3 & 0 & 2 & 2 \\ 0 & 4 & 3 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix} \tag{4.2}$$

indicating the following relationships: $x_1 \geq x_2 \geq x_3$, $x_1 \leq x_5 \leq x_6$ and (x_4, x_6) are related via an unknown relationship.

4.2 Innovization procedure

The learning agent operation (Figure 4.1) is described in Algorithm 4.1. Every variable pair (i, j) in a group G_k is cross-checked with the relationship matrix U to determine which variable pairs are designated by the user to be connected by an unknown relationship ($u_{ij} = 1$). A reference vector ($\mathbf{x}_{ref}$), calculated from the best solutions obtained so far, encodes the 'average' relationship between each variable pair among good designs found so far. The calculation procedure varies among different repair operators.

A score p_{ij} is assigned equal to the proportion of the good solution set that follow the relationship existing among variable pair (i, j), defined using $\mathbf{x}_{ref}$. This score represents how well ($\mathbf{x}_{ref}$) reflects the set of good solutions and also acts as the probability with which a particular offspring is repaired.

4.3 Repair procedure

For every variable group, if an explicit relation is defined ($u_{ij} > 1$), the repair operator ensures every offspring follows it. Otherwise, the relationships learned through innovization ($u_{ij} = 1$) are applied with the probability calculated during the innovization procedure.

The repair operators used in this study are outlined in Table 4.1. Each operator uses the knowledge available during the optimization run to different extents. For example, operator IR1 constrains the offspring the least, whereas operator IR3 constrains them the most. In addition, an ensemble operator (I-ES) is also proposed that automatically switches between IR1, IR2 and IR3

Algorithm 4.1 Learning agent algorithm

```
Require: Good solution set ($\mathcal{X}$), variable groups ($G$), relationship matrix ($U$)
Ensure: Reference vector ($\mathbf{x}_{ref}$), rule scores
 1: Calculate reference vector ($\mathbf{x}_{ref}$) from $\mathcal{X}$;        ▷ According to selected repair operator
 2: for each group $G_k$ in $G$ do
 3:     for each variable pair $(i, j)$ in $G_k$ do
 4:         $p_l \leftarrow 0$;                                   ▷ Rule score for $x_i \le x_j$
 5:         $p_g \leftarrow 0$;                                   ▷ Rule score for $x_i \ge x_j$
 6:         $p_e \leftarrow 0$;                                   ▷ Rule score for $x_i \approx x_j$
 7:         if $u_{ij} = 1$ then
 8:             for each solution $\mathbf{x}$ in $\mathcal{X}$ do
 9:                 if $x_i \approx x_j$ and $x_{i_{ref}} \approx x_{j_{ref}}$ then
10:                     $p_e \leftarrow p_e + 1$;
11:                 else if $x_i \le x_j$ and $x_{i_{ref}} \le x_{j_{ref}}$ then
12:                     $p_l \leftarrow p_l + 1$;
13:                 else if $x_i \ge x_j$ and $x_{i_{ref}} \ge x_{j_{ref}}$ then
14:                     $p_g \leftarrow p_g + 1$;
15:                 end if
16:             end for
17:             $p_{ij} \leftarrow \frac{\max(p_l, p_g, p_e)}{p_l + p_g + p_e}$;
18:         end if
19:     end for
20: end for
```

based on their individual performances. If no repair operators are used, the algorithm is referred to as 'Base MOEA' in this study.

4.3.1 MOEA/IR1 Procedure

It is assumed that the user has already provided the relationship matrix U and variable groups G. MOEA/IR1 computes the variable-wise average ($x_{i_{avg}}$) from all non-dominated (ND) solutions at the end of a generation and repairs a pair of variables of an offspring solution (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 4.1. 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 among the non-dominated (ND) solutions and to enforce repair of 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, the (x_i, x_j) pair is not modified. However, if $x_i \geq x_j$, disagreeing with the relationship found between their average ND values, the repaired (x_{ir}, x_{jr}) pair in Table 4.1 are closer to their $(x_{i_{avg}}, x_{j_{avg}})$ values. After the repair, the difference $|x_{i_{avg}} - x_{ir}| = |x_{i_{avg}} - 0.5(x_{i_{avg}} + x_i)| = 0.5|x_{i_{avg}} - x_i|$ 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 4.1) so that the supplied relationship among the two variable is satisfied. For $u_{ij} = 4$, $x_{i_r} = x_{j_r} = random(x_{i_{avg}}, x_{j_{avg}})$ is assigned.

4.3.2 MOEA/IR2 Procedure

This repair operator follows a similar repair process to 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 the past to the current generation, as shown in Table 4.1. 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 toward a projected good solution. The average value $\mathbf{x}_{avg}$ of the variables under consideration reflects the approximate pattern followed by good solutions in the current generation. The momentum term takes into account the average values in the previous generation $\mathbf{x}_{avg}(t-1)$, and allows for faster convergence toward 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 values, rather than current average value alone. The update of variables for $u_{ij} = 4$ is identical to that in IR1.

4.3.3 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 the x_i values among ND solutions of the 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

Table 4.1 Repair operator description.

Relation	Operator	Repair equations
Unknown Relationship ($u_{ij} = 1$)	IR1	$x_{i_r} = \begin{cases} \frac{x_{i_{avg}}+x_i}{2}, & \text{for } (x_{i_{avg}} - x_{j_{avg}})(x_i - x_j) < 0, \\ x_i, & \text{otherwise.} \end{cases}$ $x_{j_r} = \begin{cases} \frac{x_j+x_{j_{avg}}}{2}, & \text{for } (x_{i_{avg}} - x_{j_{avg}})(x_i - x_j) < 0, \\ x_j, & \text{otherwise.} \end{cases}$
	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 $x_{i_{ref}}(t) = x_{i_{avg}}(t) + \gamma \left(x_{i_{avg}}(t) - x_{i_{avg}}(t-1)\right)$
	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]
Direct Relationship ($u_{ij} = [2,3]$)	IR1	$x_{i_r} = \begin{cases} x_{ij_{avg}} - \frac{\lvert x_{ij_{avg}} - x_{i_{avg}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{avg}} + \frac{\lvert x_{ij_{avg}} - x_{i_{avg}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ $x_{j_r} = \begin{cases} x_{ij_{avg}} + \frac{\lvert x_{ij_{avg}} - x_{j_{avg}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{avg}} - \frac{\lvert x_{ij_{avg}} - x_{j_{avg}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ where $x_{ij_{avg}} = \frac{x_{i_{avg}} + x_{j_{avg}}}{2}$.
	IR2	$x_{i_r} = \begin{cases} x_{ij_{ref}} - \frac{\lvert x_{ij_{ref}} - x_{i_{ref}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{ref}} + \frac{\lvert x_{ij_{ref}} - x_{i_{ref}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ $x_{j_r} = \begin{cases} x_{ij_{ref}} + \frac{\lvert x_{ij_{ref}} - x_{j_{ref}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{ref}} - \frac{\lvert x_{ij_{ref}} - x_{j_{ref}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ where $x_{i_{ref}}(t) = x_{i_{avg}}(t) + \gamma \left(x_{i_{avg}}(t) - x_{i_{avg}}(t-1)\right)$
	IR3	$x_{i_r} = \begin{cases} x_{ij_{ref}} - \frac{\lvert x_{ij_{ref}} - x_{i_{ref}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{ref}} + \frac{\lvert x_{ij_{ref}} - x_{i_{ref}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ $x_{j_r} = \begin{cases} x_{ij_{ref}} + \frac{\lvert x_{ij_{ref}} - x_{j_{ref}} \rvert}{2}, & \text{for } u_{ij} = 2, \\ x_{ij_{ref}} - \frac{\lvert x_{ij_{ref}} - x_{j_{ref}} \rvert}{2}, & \text{for } u_{ij} = 3. \end{cases}$ where $x_{i_{ref}}(t) = x_{i_{avg}}(t) + \gamma \left(x_{i_{avg}}(t) - x_{i_{avg}}(t-1)\right)$

identical to that in IR1 and IR2.

4.3.4 Ensemble MOEA/I-ES Procedure

All repair operators designed for MOEA/I must 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 offspring generated by each. This relieves the need to analyze every operator, since the best performing will be selected automatically. The ensemble method also treats Base MOEA as a repair operator, which represents the case when the offspring is not repaired. This allows the optimization to remain relatively 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 the offspring created by an operator, greater is the probability of it being used to repair subsequent offsprings. The probability ($\widehat{p_r^i}$) update operation for the i-th operator is described below:

$$p_r^i(t+1) = \max\left(p_{min},\ \alpha\frac{r_s^i}{\sum_i r_s^i} + (1-\alpha)\widehat{p_r^i}(t)\right), \tag{4.3}$$

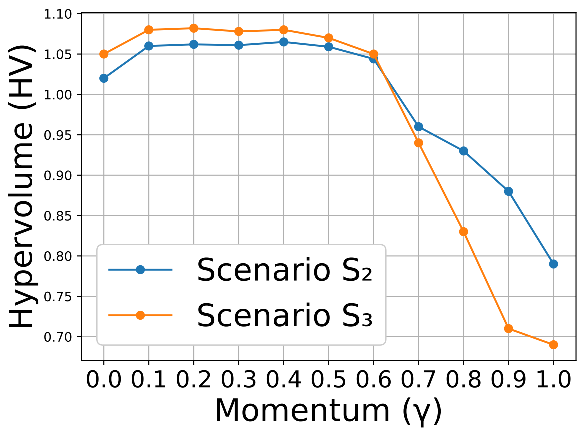
$$\widehat{p_r^i}(t+1) = \frac{p_r^i(t+1)}{\sum\limits_i p_r^i(t+1)}, \tag{4.4}$$

where α is the learning rate, $r_s^i = \frac{n_s^i}{n_{off}}$ (n_s^i and n_{off} are the number of offsprings created by the i-th operator that survive in generation t and the total number of offsprings that survive in generation t, respectively). It is possible that at any point during the optimization, no solution generated by one of the repair operators survives. This might cause the corresponding selection probability to go down to zero without any possibility of recovery. To prevent this, in Equation 4.3, the probability update step ensures that a minimum selection probability (p_{min}) is always assigned to each repair operator present in the ensemble. Equation 4.4 normalizes the probability values for each operator so that their total sum is 1.

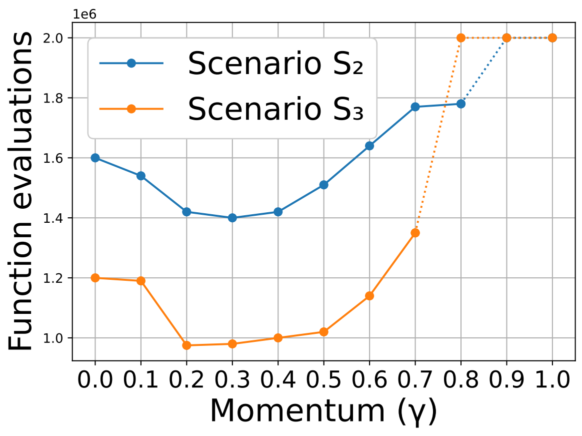
The learning rate (α) determines the rate of change of the repair probabilities. A high α would increase the sensitivity, and can result in large changes in repair probabilities over a short period of time. A low α exerts a damping effect which causes the probability values to update slowly. Through trial and error, $\alpha = 0.5$ and $p_{min} = 0.1$ were found to be suitable to the problems of this study.

4.4 Sensitivity analysis of momentum parameter

For the IR2 and IR3 repair operators, the calculation of $x_{i_{ref}}$ requires specifying a momentum parameter (γ). A sensitivity analysis of γ is presented in this section. For scenarios S_2 and S_3 of the 820-member truss, γ was set at different values between 0 to 1, and 20 runs were performed for each setting using NSGA-II/IR2. The median HV values obtained after 2 million function evaluations are recorded for each γ setting. The results are outlined in Figure 4.2.



(a) Median HV after 2M fevals.



(b) Median fevals taken to reach HV^T=0.91.

Figure 4.2 Performance of NSGA-II/IR2 for each $\gamma = 0, 0.1, 0.2, ..., 1$. For each γ combined with scenarios S_2 and S_3, 20 runs were performed.

From Figure 4.2a, it is evident that the HV variation is very low for $\gamma <= 0.6$, whereas for higher values the median final HV sharply drops. In Figure 4.2b, it is seen that $0.2 \leq \gamma \leq 0.4$ results in lower function evaluations required to achieve HV^T. The difference in the number of function evaluations is also low with this setting, indicating a low parameter sensitivity in this region. The dotted lines signify that NSGA-II/IR2 was not able to reach HV^T within 2 million function evaluations. Taking the results of both the HV and function evaluations sensitivity analysis into account, γ is set to be 0.3 in order to ensure maximum performance and low parameter sensitivity.

4.5 Truss Design Problem

4.5.1 Problem background and formulation

Truss structures provide a wide variety of optimization applications [94, 95, 96, 97], hence, a number of algorithms have been published to solve them. Genetic algorithms (GAs) have been applied to truss design problems by [98, 99] for single objective, [100, 101] for multiple objectives, and [102] for many objectives.

For this study, a scalable 3D truss design problem has been created based on [92, 103]. One such truss having 260 cylindrical members is shown in Fig. 4.3. 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 each of the top nodes as shown in Fig. 4.3. The number of truss members can be increased as necessary to scale up the problem.

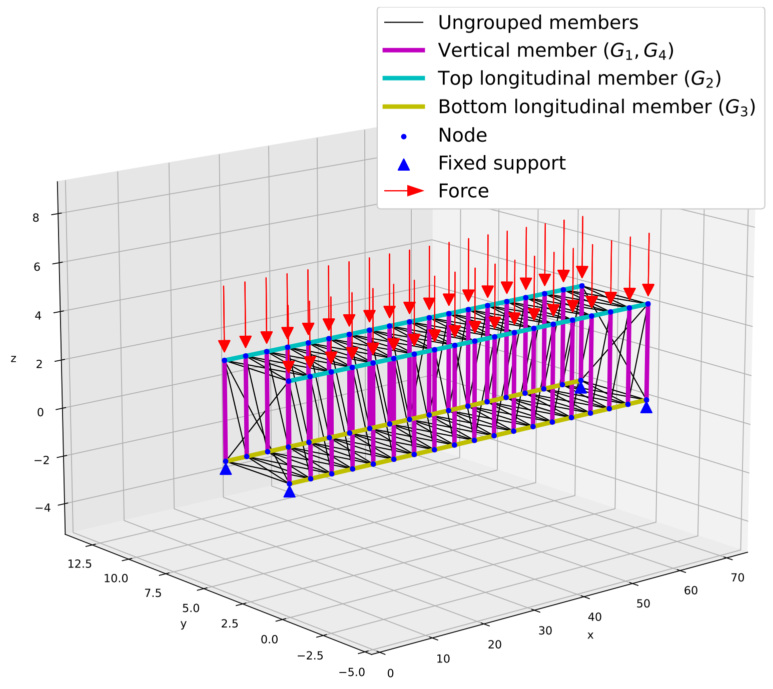

Ungrouped members
Vertical member (G_1, G_4)
Top longitudinal member (G_2)
Bottom longitudinal member (G_3)
Node
Fixed support
Force
z
y
x

Figure 4.3 A truss structure with support and vertical loadings.

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 ($\mathbf{L}_v$) parallel to the z-axis. The $\mathbf{L}_v$ for the vertical members at $y = 0$ is assumed to be the same as that for those at $y = 4$. This prevents too much arbitrariness in the truss shape. This problem can be scaled by adjusting the number of shape variables (n_s). Two constraints are to be satisfied: (a) stress in 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:

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{r}, \mathbf{L}_v) = \rho \sum_{i=1}^{n_m} V_i(\mathbf{r}, \mathbf{L}_v), && (4.5)\\
\text{Minimize} \quad & f_2(\mathbf{r}, \mathbf{L}_v) = \mathbf{F}(\mathbf{r}, \mathbf{L}_v) \cdot \mathbf{D}(\mathbf{r}, \mathbf{L}_v), && (4.6)\\
\text{Subject to} \quad & \sigma_i(\mathbf{r}, \mathbf{L}_v) \leq S, \quad \text{for } i = 1, 2 \ldots, n_m, && (4.7)\\
& |D_j(\mathbf{r}, \mathbf{L}_v)| \leq \delta, \quad \text{for } j = 1, 2, \ldots, n_n, && (4.8)\\
& r^L \leq r_i \leq r^U, \ \text{for } i = 1, 2 \ldots, n_m, && (4.9)\\
& l^L \leq l_j \leq l^U, \ \text{for } j = 1, 2 \ldots, n_n, && (4.10)
\end{aligned}
$$

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 $\mathbf{D}$ are vectors consisting of the forces in members and displacements at nodes, respectively. σ_i is the stress developed in the i-th member of 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 evaluations require a finite element simulation, making the process computationally expensive. 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.

4.5.2 User-provided information

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. In this problem, the radii of members are comparable variables. The same is true for the lengths of vertical members. Thus, a pair of variables (i, j) within each of these classes (radii and lengths) can be be assigned a relationship index $u_{ij} > 0$. 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. 4.3, 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, the length of vertical members will monotonically increase while moving 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 radii of the corresponding members may also monotonically increase to withstand the large bending moment.

The incorporation of this *a priori* user knowledge into the optimization process requires defining the variable groups (G) and the relationship matrix (U) defining the associated relationships. Four groups of variables are considered in this study, as shown in Table 4.2. The truss members corresponding to each group are also highlighted in Fig. 4.3. Each group is also divided into two or more subgroups. 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 to a member with a different location and physical orientation.

Based on the variable groups, eight different scenarios (S_1 to S_8) are created with varying extents of user knowledge being supplied, as shown in Table 4.3. Each scenario uses the available groups and subgroups in different ways. Each entry in the table represents the values (u_{ij}) constituting the matrix U for every variable pair $(i, j) \in G_k$. The first four scenarios from S_1 to S_4 do not

Table 4.2 Variable groups for the 820 and 1,380-member truss cases. Each group has comparable variables having identical units and scales.

Group	Subgroup	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}		[30 – 59]	[50 – 99]
G_2	G_{21}	r_i of top longitudinal members	[60 – 88]	[60 – 108]
	G_{22}		[89 – 117]	[109 – 157]
	G_{23}		[118 – 146]	[158 – 206]
	G_{24}		[147 – 175]	[207 – 255]
G_3	G_{31}	r_i of bottom longitudinal members	[176 – 204]	[256 – 304]
	G_{32}		[205 – 233]	[305 – 353]
	G_{33}		[234 – 262]	[354 – 402]
	G_{34}		[263 – 291]	[403 – 451]
G_4	G_{41}	r_i of vertical members	[292 – 321]	[452 – 501]
	G_{42}		[321 – 350]	[501 – 550]
	G_{43}		[351 – 380]	[551 – 600]
	G_{44}		[380 – 409]	[600 – 649]

enforce any symmetry, making 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 subgroups G_{11} and G_{12}. This signifies that the length of each vertical member is expected 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 subgroup 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 to various degrees dictated by the relative success of the three repair schemes.

Scenario S_4 provides more problem information to the optimization algorithm by specifying precise relationships among variables of each subgroup of the four groups. For example, within the

Table 4.3 User information 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. For example, for S_6 and G_1, $u_{ij} = 4$ means $r_i \approx r_{60-i}$ for $i = 1, \ldots, 30$. For S_7 and G_2, $u_{ij} = 4$ means $r_i \approx r_{177-i}$ where $i = 60, \ldots, 117$, for subgroups G_{21} and G_{22}.

Group	Subgroup	Scenario							
		S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
G_1	G_{11} G_{12}	0	1 1	1 1	2 3	4	1 4 1	1 4 1	2 4 3
G_2	G_{21} G_{22}	0	0	1 1	2 3	0	0	1 4 1	2 4 3
	G_{23} G_{24}	0	0	1 1	2 3	0	0	1 4 1	2 4 3
G_3	G_{31} G_{32}	0	0	1 1	2 3	0	0	1 4 1	2 4 3
	G_{33} G_{34}	0	0	1 1	2 3	0	0	1 4 1	2 4 3
G_4	G_{41} G_{42}	0	0	1 1	2 3	0	0	1 4 1	2 4 3
	G_{43} G_{44}	0	0	1 1	2 3	0	0	1 4 1	2 4 3

subgroup 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}, ($u_{i,i+1} = 3$): $x_i \geq x_{i+1}$ 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 the subgroups 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.

In this study, only inequality-based relationships are considered 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 until the final scenario, which specifies the exact relationships expected to be present in optimal solutions.

4.5.3 Experimental Settings

In this study, NSGA-II [15], 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 4.3.1 to 4.3.4, 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, described in Section 4.5.3.1. The parameter settings for NSGA-II as well as for the repair operators are presented in Table 4.4. Two truss cases, one with 820 members, and another with 1,380 members, were considered. The variable groups were set according to Table 4.2. Different experiments were performed with multiple levels of user knowledge integration, which are described in Section 4.5.2. 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 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 for both truss cases, the maximum number of function evaluations available was set at 2 million.

The hypervolume (HV) metric [104, 105, 106] 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). Over 20 runs, for each method, the mean and standard deviation of the number of function evaluations taken to achieve HV^T are used for performance comparison. The Wilcoxon rank-sum test [107, 108] 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 performance metric values for two algorithms A_1 and A_2. For each run, x_1 and x_2 exist 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 not a statistically significant performance difference between A_1 and A_2. In this case, the algorithm with the lowest number of median function evaluations ($\text{FE}^{\text{min}}_{median}$) required for achieving HV^T is chosen as reference. The HV values for achieving one standard deviation more than lowest median FE ($\text{FE}^{\text{min}}_{median}+\sigma^{\text{min}}_{FE}$) for all algorithms are recorded for 20 runs. Using this data, the Wilcoxon test is performed and the p-values are calculated.

Table 4.4 Parameter settings of NSGA-II/I.

Parameter	Value
Population size	500
Maximum generations	4,000
Mutation operator	Polynomial mutation [11]
Mutation probability (p_m) and index (η_m)	$1/n_{var}$, 50
Crossover operator	SBX [3]
Crossover probability (p_c) and index (η_c)	0.9, 30
Momentum parameter (γ) for IR2	0.3
Minimum repair prob. (p_{min}) in Equation 4.3	0.1
α in Equation 4.3	0.5

4.5.3.1 Comparison with semi-independent variables

Scenarios S_4 and S_8 explicitly define the relationships to be imposed 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, methods such as semi-independent variables (SIVs, v_i) proposed by [6] can be used to enforce the provided knowledge as constraints. In this study, this algorithm is called NSGA-II/SIV. For example, for subgroup G_{11} with K_{11} variables under scenario S_4 or S_8 defined in Table 4.2, the SIV representation is shown below. A similar process can be repeated for the other variable groups as well.

Specified relationship: $l_1 \leq l_2 \leq \ldots \leq l_{K_{11}}$,

SIVs: l_1 (base var.), $v_2, v_3, \ldots, v_{K_{11}}$ (derived vars.),

where

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

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 $u_{i,i+1} = 3$.

4.5.4 Results and Discussion

The optimization results for the 820 and 1,380-member truss cases are presented in Tables 4.5 and 4.6, respectively. 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 unknown 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:

$$\mathcal{K} = \frac{\sum\limits_{i<j} [u_{ij} = 4] + 0.75 \sum\limits_{i<j} [u_{ij} = 2 \vee 3] + 0.5 \sum\limits_{i<j} [u_{ij} = 1]}{n(n-1)/2}. \tag{4.11}$$

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 Base NSGA-II. This is because S_1 is the no-knowledge case where all the elements in the matrix U are set to 0. Thus, all the repair operators remain inactive, and the results are the same as Base NSGA-II. For most of the cases, it is seen that NSGA-II/IR2 performs the best (marked in bold), except for scenarios S_5 and S_8 (both cases), and S_4 (only for 1380-member truss). For S_5, even though Base NSGA-II performs the best in terms of FEs, the other algorithms give a statistically similar performance ($p > 0.05$). For S_8, all the repair operators and NSGA-II/SIV have a statistically worse performance ($p < 0.05$) compared

Table 4.5 FEs (median and standard deviation) required to reach target $\text{HV}^T = 0.91$ for 820-member, 236-node truss problem. Best performing algorithm for each scenario (S_1 to S_4 does not use symmetry of any kind but more knowledge is supplied from S_1 towards S_4, and similar knowledge embedding is done for S_5 to S_8, except that symmetry knowledge is used for these latter scenarios) is marked in bold. Algorithms with performance not statistically different from the best algorithm are marked in italics. N/A indicates 'Not Applied' when the user-provided information is not used (Base NSGA-II), or used in specific instances (NSGA-II/SIV). Wilcoxon test p-values are given in braces.

Scn.	$\mathcal{K}$	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*±65k (p =0.1343)	**1.4M**±72k	*1.7M*±9k (p =0.0951)	*1.5M*±50k (p =0.1520)	N/A
S_3	3.80e-04	2M (p =0.0213)	*1.2M*±30k (p =0.1830)	**0.98M**±33k	*1.5M*±25k (p =0.0961)	*1.2M*±27k (p =0.1841)	N/A
S_4	5.60e-04	*2M* (p =0.0424)	1.9M±23k (p =0.0411)	**1.7M**±20k	2M (HV=0.52, p = 0.0068)	*1.8M*±10k (p =0.1313)	*1.9M*±20k (p =0.0313)
S_5	6.50e-04	**1.2M**±8k	*1.3M*±18k (p =0.1363)	*1.3M* ±12k (p =0.1413)	*1.4M*±8k (p =0.1526)	*1.3M*±21k (p =0.1201)	N/A
S_6	7.60e-04	*1.2M*±8k (p =0.0414)	*0.99M*±22k (p =0.1328)	**0.96M**±25k	*1.2M*±22k (p =0.0983)	*0.98M*±21k (p =0.1154)	N/A
S_7	1.16e-03	1M±10k (p =0.0048)	*0.72M*±19k (p =0.0612)	**0.44M**±23k	0.92M±10k (p =0.0141)	*0.65M*±15k (p =0.0596)	N/A
S_8	1.30e-03	**1M**±10k	*1.3M*±21k (p =0.0362)	1.6M±30k (p =0.0419)	1.8M±20k (p =0.0261)	1.5M±27k (p =0.0237)	1.8M±10k (p =0.0225)

to Base NSGA-II. Tables 4.5 and 4.6 show that a moderate usage of knowledge obtained through innovization gives the optimal results. Too little or too much knowledge is detrimental to the optimization performance. The ensemble approach's (NSGA-II/I-ES) performance is statistically similar to the best algorithm in all scenarios except S_8. For example, with a Wilcoxon ranksum test, it is observed that for S_7 in Table 4.5, NSGA-II/I-ES has a statistically similar performance (with $p = 0.0596$) to NSGA-II/IR2, the best-performing algorithm. The ensemble approach (NSGA-II/I-ES) has the advantage that the user does not need to think about which repair operator to use, since it is handled automatically.

Figure 4.4 shows the variation of solution evaluations for different algorithms and scenarios, compiled from Table 4.5. It is clear from both plots (with and without the use of symmetry knowledge) that the best performance is achieved with an intermediate supply of problem knowledge

(S_3 and S_7) combined with an intermediate usage extent (IR2).

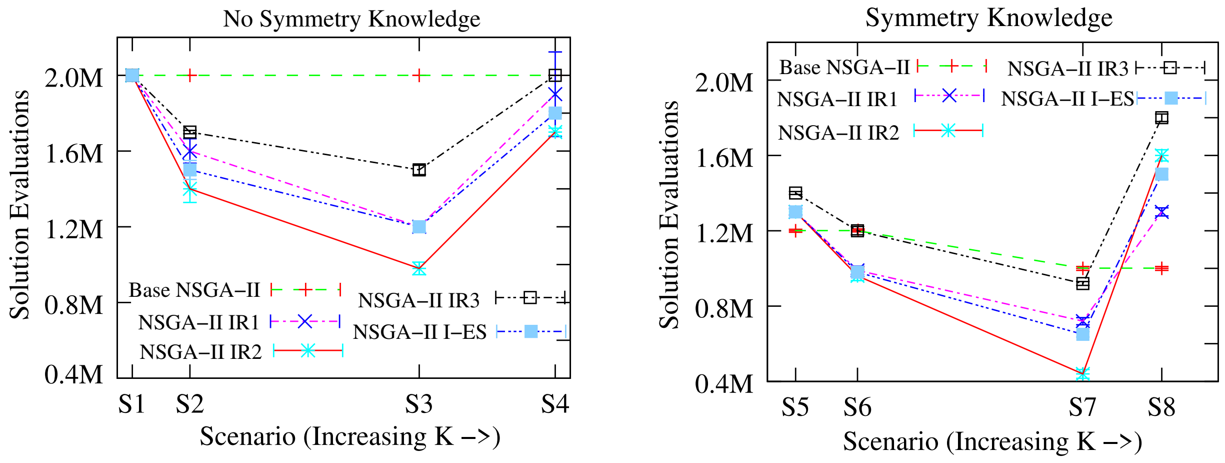


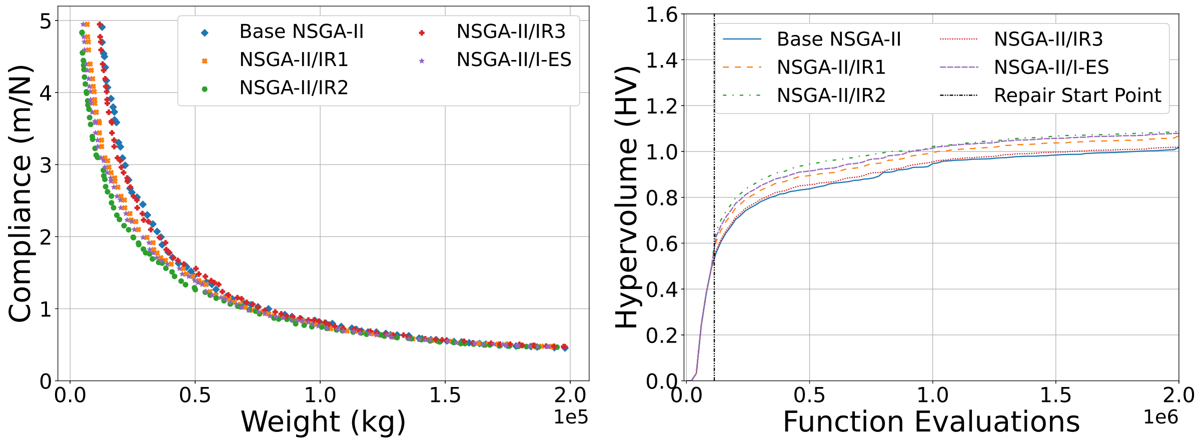
Figure 4.4 Variation of solution evaluations for different algorithms and different scenarios for 820-member, 236-node truss problem.

Proceeding row-wise from top to bottom, it is seen that performance generally improves until 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. An interesting comparison is the difference in performances between NSGA-II/IR approaches and NSGA-II/SIV [6] (applicable to cases with $u_{ij} = 2$ or 3 only). For the 820-member truss case in Table 4.5, the performance of NSGA-II/SIV is close to NSGA-II/I-ES and is 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 (Table 4.6), NSGA-II/SIV is not able to reach the desired performance within 2M function evaluations for all the scenarios. This shows that the SIV approach is still susceptible to knowledge overspecification for large-sized problems.

Results for both trusses for scenario S_7 are shown in Figures 4.5 and 4.6, respectively. The ND front plots (Figures 4.5a and 4.6a) clearly show NSGA-II/IR2 as the better performer, providing higher quality solutions than the others, with NSGA-II/I-ES following close behind. The median

Table 4.6 FEs required to reach target HV^T = 0.80 for 1,380-member, 396-node truss problem.

Scn.	$\mathcal{K}$	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*±42k (p =0.3212)	**1.6M**±48k	*1.8M*±9k (p =0.3205)	*1.7M*±25k (p =0.3153)	N/A
S_3	4.70e-03	2M (p =0.0153)	*1.6M*±20k (p =0.1394)	**1.4M**±25k	1.9M±10k (p =0.0265)	*1.5M*±17k (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**±25k	*1.7M*±15k (p =0.3120)	*1.7M*±23k (p =0.2961)	*1.8M*±12k (p =0.1849)	*1.7M*±11k (p =0.2971)	N/A
S_6	3.26e-02	1.6M±25k (p =0.0431)	*1.3M*±20k (p =0.0713)	**1.0M**±30k	1.5M±10k (p =0.0481)	*1.1M*±18k (p =0.0692)	N/A
S_7	3.59e-02	1.4M±15k (p =0.0303)	*1.2M*±19k (p =0.0572)	**0.90M**±23k	*1.2M*±10k (p =0.0576)	*1.0M*±15k (p =0.0861)	N/A
S_8	4.83e-02	**1.4M**±15k	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)

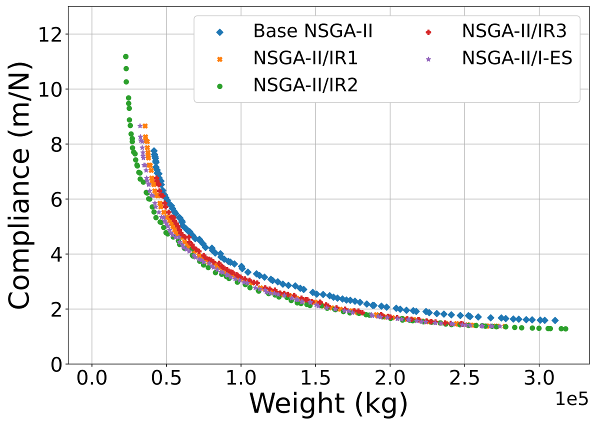


(a) ND fronts. (b) Median hypervolume of 20 runs.

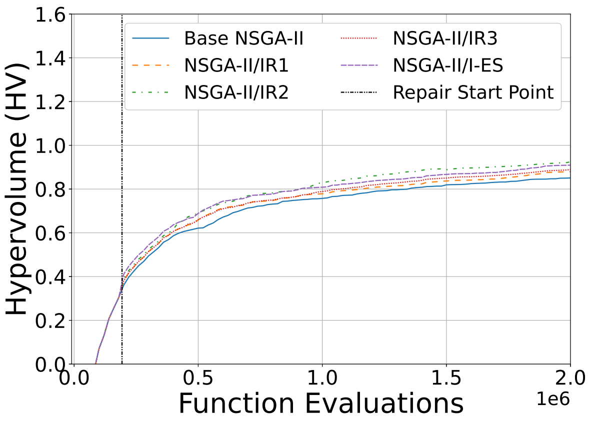
Figure 4.5 Results for 820-member truss for scenario S_7 (symmetry among subgroups, unknown relationships within subgroups).

HV plots (Figures 4.5b and 4.6b) show that NSGA-II/IR2 and NSGA-II/I-ES achieve a higher median HV, faster, as shown previously in Tables 4.5 and 4.6.

The NSGA-II/I-ES median repair probabilities plots (Figure 4.7) show 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

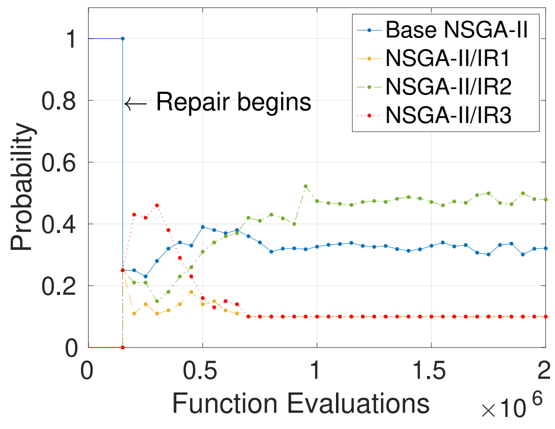


(a) ND fronts.

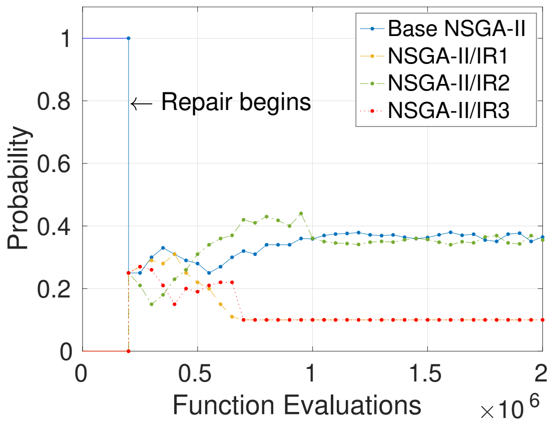


(b) Median hypervolume of 20 runs.

Figure 4.6 Results for 1,380-member truss for scenario S_7 (symmetry among subgroups, unknown relationships within subgroups).



(a) 820-member truss.



(b) 1,380-member truss

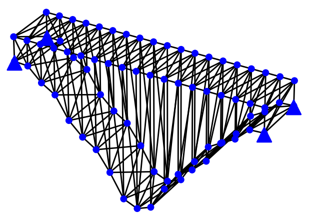
Figure 4.7 Repair probability variation for median run.

of IR1 and IR3 are reduced to the minimum level very early on, and new solution generation is controlled mostly by IR2 and the base NSGA-II. This clearly indicates the ability of the proposed ensembled method (I-ES) to pick the most successful operator on the fly for solving a problem efficiently.

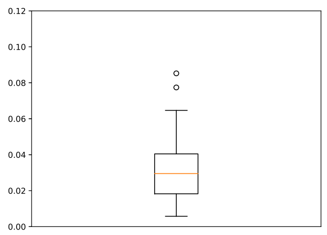
It is interesting from Tables 4.5 and 4.6 that with more information ($\mathcal{K}$) being provided, the performance of all algorithms does not improve monotonically. This means that there exists an optimal amount of problem information for every problem below or above which there is under- or over-specification of 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 in searching for 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 MOEA to have a flexible search, thereby requiring to use an minimal number of solution evaluations to solve a problem. For both trusses, the S_7 scenario provides the right amount of additional problem information. Combining it with the NSGA-II/IR2 algorithm which uses the right amount of available knowledge, the problem can be solved with at most 22% and 45% of the median number of solution evaluations required by the base NSGA-II procedure, for the 820- and 1,380-member problems, respectively. NSGA-II/I-ES provides a statistically comparable performance, thus negating the need to know the appropriate repair operator in advance.

4.5.4.1 Visualization of non-dominated solutions

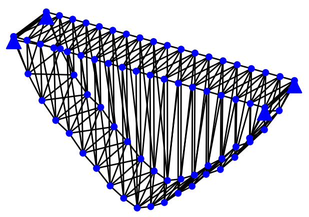
In this section, details about some of the best non-dominated solutions for the two truss case studies have been provided. For each truss case, three designs taken from the Pareto Front of one run for NSGA-II/IR2 for Scenario S_7 have been visualized in Figures 4.8 and 4.9. The first two designs in each figure are the lowest weight and lowest compliance trusses, thus covering the two extremes of the Pareto front. The third design is taken from the middle of the Pareto front. The member size distribution for each design is also provided.



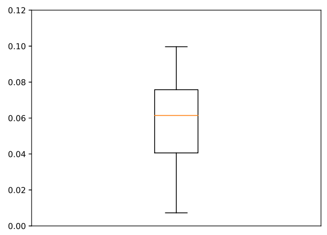
(a) Min. weight truss
(4,000 kg, 5.0 m/N)



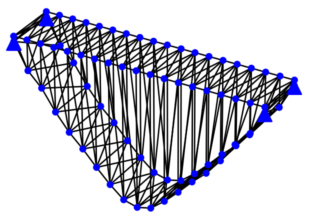
(b) Member size distribution
(Median = 0.03 m^2)



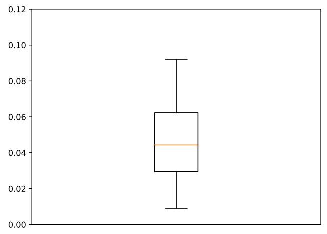
(c) Min. compliance truss
(200,000 kg, 0.5 m/N)



(d) Member size distribution.
(Median = 0.06 m^2)

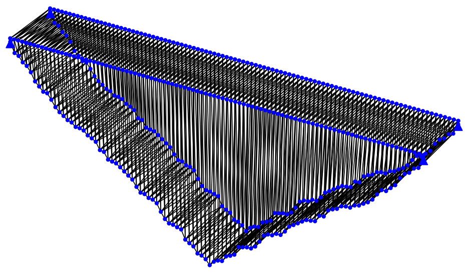


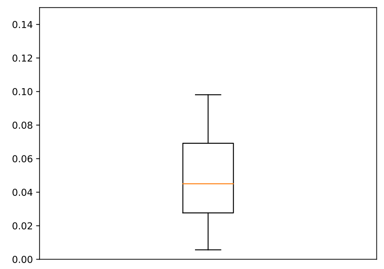
(e) Intermediate truss
(102,000 kg, 0.8 m/N)



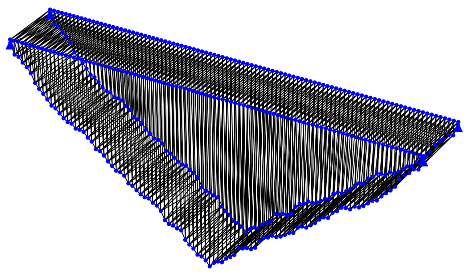
(f) Member size distribution.
(Median = 0.04 m^2)

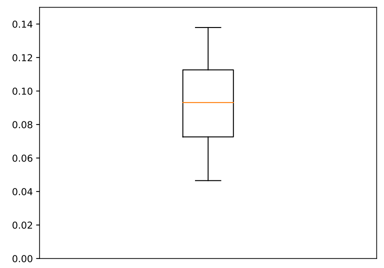
Figure 4.8 Figures (a), (c) & (e) show 3 truss topologies found in one run of NSGA-II/IR2 for the 820-member truss problem and scenario S_7. The first two trusses correspond to the two extremes of the Pareto Front (minimum weight and compliance). The third truss represents a solution picked from the middle region of the Pareto Front. Figures (b), (d) & (f) show the variation in member sizes among the 3 trusses.


(a) Min. weight truss
(24,000 kg, 11.8 m/N)

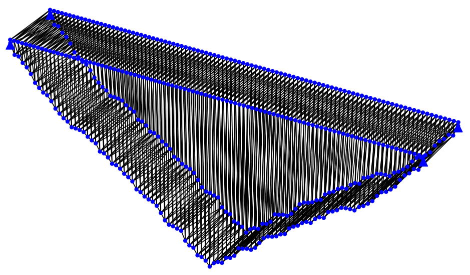


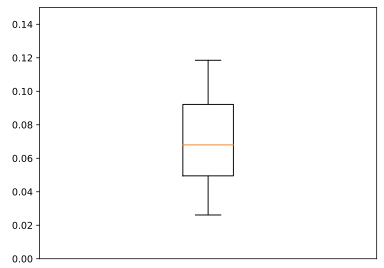
(b) Member size distribution
(Median = 0.04 m^2)


(c) Min. compliance truss
(325,000 kg, 1.7 m/N)



(d) Member size distribution.
(Median = 0.09 m^2)


(e) Intermediate truss
(175,000 kg, 1.9 m/N)



(f) Member size distribution.
(Median = 0.07 m^2)

Figure 4.9 Figures (a), (c) & (e) show the optimal truss designs found in one run of NSGA-II/IR2 for the 1380-member truss problem and scenario S_7. Figures (b), (d) & (f) show the variation in member sizes among the 3 trusses.

4.6 Solid Rocket Design Problem

4.6.1 Problem background and formulation

Optimal design of solid rocket motors is a pertinent problem due to the advantages of solid fuels ([109]). 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 heterogeneous propellant distribution problem can be formulated as an optimization problem as done by [91] and the same formulation is used in this study. 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. 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 propellants 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.

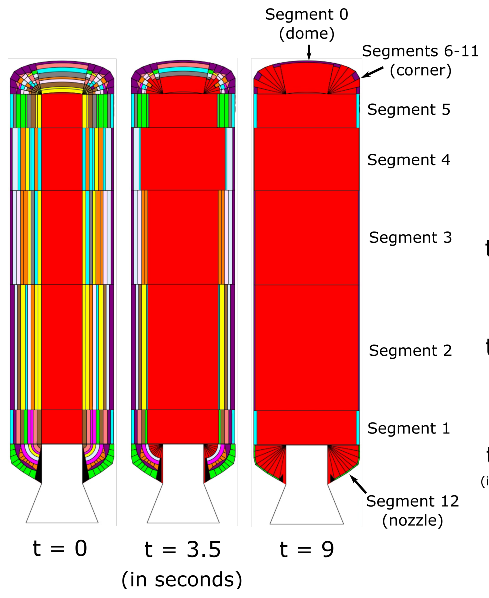
The rocket under consideration has the propellant regions divided vertically into six cylindrical sections, plus the dome, with each having a maximum of 20 distinct layers arranged as concentric rings. Three sections possess some additional propellant arranged at the core, also referred to as a 'star' or finocyl 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. Propellant types and their burn rates are presented in Table 4.7.

Figures 4.10a and 4.10b illustrate the burn progression for a specific propellant arrangement.

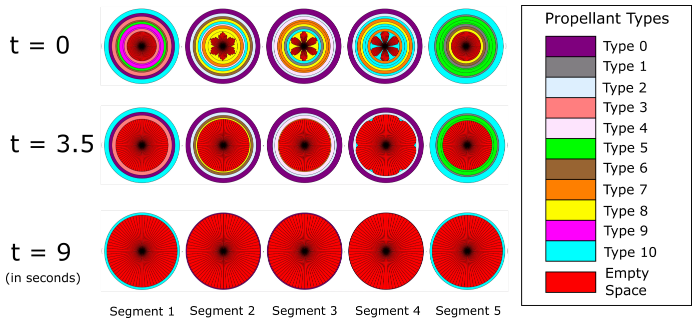
The target thrust profile is defined as a monotonically decreasing curve defined at 0.5-second intervals, and lasting for 10 seconds, as shown by the blue band in Figure 4.12a. A good design

Table 4.7 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



(a) Side view.



(b) Sectional views at different segments.

Figure 4.10 Side and top views of a rocket showing burn progression from ignition till burnout. 10 propellant types with varying burn rates are represented by separate colors. The side view only shows the propellants arranged in cylindrical sections. The middle 3 segments for the top view at t=0 are the ‘star’ segments with a non-cylindrical geometry initially. Adapted from [91].

should match the target thrust profile with a tolerance of ±5%.

The optimization problem formulation is given below:

$$\text{Minimize} \quad f_1(\mathbf{x}) = \sum_{t=0}^{t_b} (T(\mathbf{x}, t) - T_r(t))^2, \tag{4.12}$$

$$\text{Minimize} \quad f_2(\mathbf{x}) = \mu_{res}(\mathbf{x}) + \sigma_{res}(\mathbf{x}), \tag{4.13}$$

$$\text{subject to} \quad P^L \leq P(\mathbf{x}, t) \leq P^U, \tag{4.14}$$

where $\mathbf{x}$ is the vector of decision variables specifying each type of propellant in each layer of each segment, thickness of each layer, and geometry of core (finocyl) 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. The allowable pressure range is set to be between 1.4 MPa and 3.5 MPa.

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 4.8.

Table 4.8 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 Equation (4.12) 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 ±5% of the target thrust.

The second objective function defined by Equation (4.13) 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 at least 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 measure the disparity in the residues across different segments. Minimizing both μ_{res} and σ_{res} is necessary for obtaining a good design.

Equation (4.14) represents the pressure constraint which is necessary to ensure optimal rocket operation. Dropping below the minimum pressure (Pressure Deflagration Limit, or PDL) will cause the rocket to irrevocably cease burning, whereas, exceeding the maximum pressure risks explosion of the rocket.

4.6.2 User Knowledge

Unlike the truss design problem which is very well-studied in literature, the solid rocket design problem is relatively new [91]. 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 4.9. 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 segment.

Five scenarios can be designed based on different levels of user knowledge extent, shown in Table 4.10. 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 [91]. 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 4.9.

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 is supplied. The target thrust profile in Figure 4.12a 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}. However, due to the problem complexity, such a trend is not guaranteed to hold for every case. Such knowledge can be potentially 'imperfect' or 'incomplete'. Compared to the other scenarios, C_5 has the most user-supplied information.

Table 4.9 Variable groups for the rocket problem. Each group has comparable variables with identical units and scales.

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 4.10 User information 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

4.6.3 Results and Discussion

The experimental settings are same as those for the truss design problem described in Section 4.5.3, with only the maximum generations increased to 100,000. For scenario C_5 consisting of explicitly-specified information, NSGA-II/SIV is also compared to the other repair operators (IR1, IR2, IR3 and I-ES).

The optimization results are presented Table 4.11. 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 all the scenarios, the p-values of the Wilcoxon test with the best-performing algorithm as a reference are also given in brackets. For the no-knowledge case C_1, all the elements of matrix 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. Scenario C_4 combined with the NSGA-II/IR2 operator gives the best performance overall. This shows that for both the amount of user-supplied knowledge and the extent of online knowledge usage, an optimal level exists. Too little or too much knowledge usage is detrimental to the optimization performance. Interestingly, for C_5 the relationships between the propellants in the cylindrical layers are explicitly defined. But this seems to constrain the algorithm and degrades its 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.

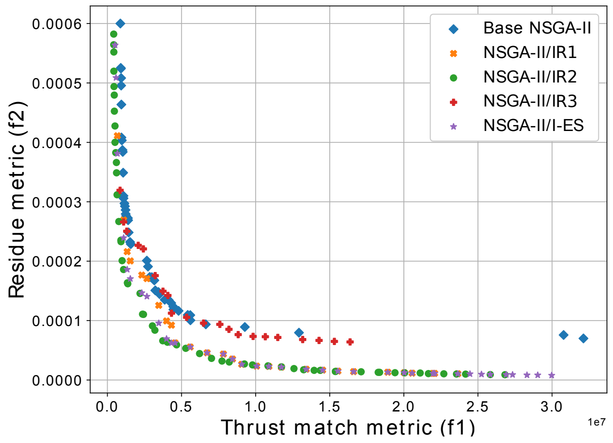
In terms of final HV obtained, for all the cases, NSGA-II/IR2 provides the fastest convergence, demonstrated by the low number of function evaluations taken to reach the target HV, with NSGA-II/I-ES following closely. Base NSGA-II always performs worse in scenarios C_2 to C_4. For C_5, the perceived 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 fronts obtained by the four repair operators for scenario C_4 is shown in Figure 4.11a. It can be seen that NSGA-II/IR2 provides better quality solutions than the others. The median HV plot in Figure 4.11b 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

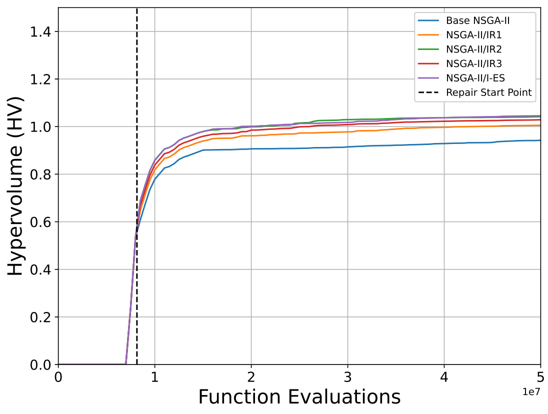
Table 4.11 FEs required to reach target HV^T = 0.93 for 544-variable solid fuel rocket design.

Scn.	$\mathcal{K}$	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*±2.0M (p =0.1205)	**30.5M**±1.5M	36.0M±3.0M (p =0.0104)	*31.5M*±1.0M (p =0.1516)	N/A
C_3	4.0e-03	50.0M (p =0.0061)	*28.0M*±3.0M (p =0.0998)	**25.0M**±4.0M	32.0M±1.0M (p =0.0051)	*26.0M*±1.0M (p =0.1336)	N/A
C_4	7.8e-03	50.0M (p =0.0029)	19.0M±5.0M (p =0.0350)	**12.0M**±5.0M	22.0M±4.0M (p =0.0154)	*13.0M*±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=0.78, p = 0.0135)	50.0M (HV=0.66, p = 0.0104)	50.0M (HV=0.84, p = 0.0157)	50.0M (HV=0.69, p =0.0119)

profile and residues for one of the solutions on the ND front is shown in Figures 4.12a and 4.12b, respectively. Thrust is always within allowable ranges and the residue for every segment is lower than 1 mm.



(a) ND fronts.

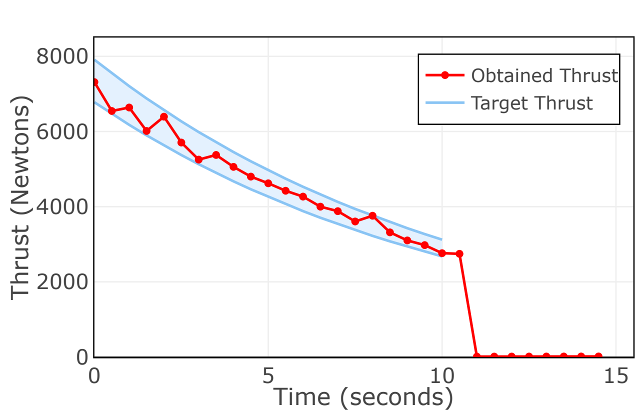


(b) HV variations.

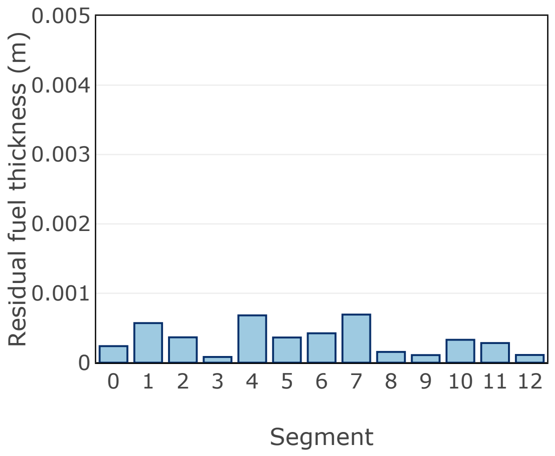
Figure 4.11 ND fronts and HV variations of scenario C_4 for rocket design.

4.6.3.1 Visualization of non-dominated solutions

Three non-dominated solutions from different regions of the Pareto Front found in one run of NSGA-II/IR2 for scenario C_4 are visualized in Figures 4.13, 4.14, and 4.15. Two types of decision variables are visualized: the ‘star’ geometry, and the cylindrical segment propellants. One common pattern visible in all 3 propellant distribution figures is the presence of faster burning fuel (reddish regions) in the inner layers and slower burning propellants (bluish regions) in the outer layers. This



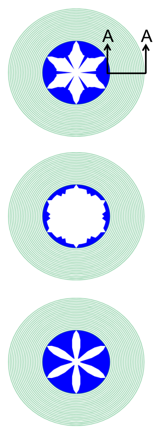
(a) Thrust profile is within limits (f_1).



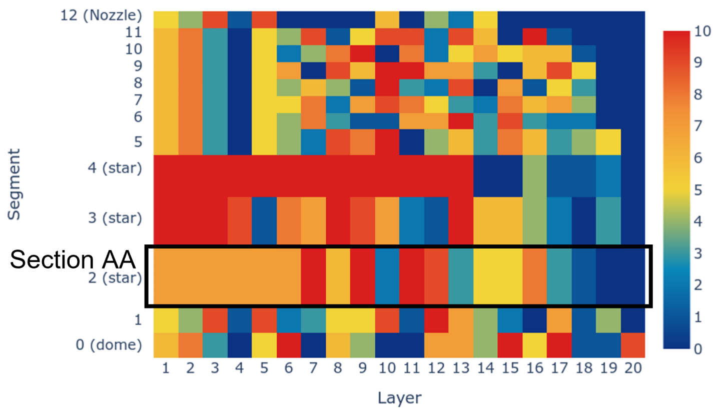
(b) Residue (f_2) is less than 1 mm.

Figure 4.12 Objectives f_1 and f_2 for a particular run for scenario C_4.

intuitively makes sense since the target thrust profile requires a higher thrust early on, thus requiring a faster burning fuel compared to the later part of the burn.

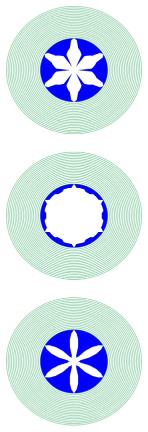


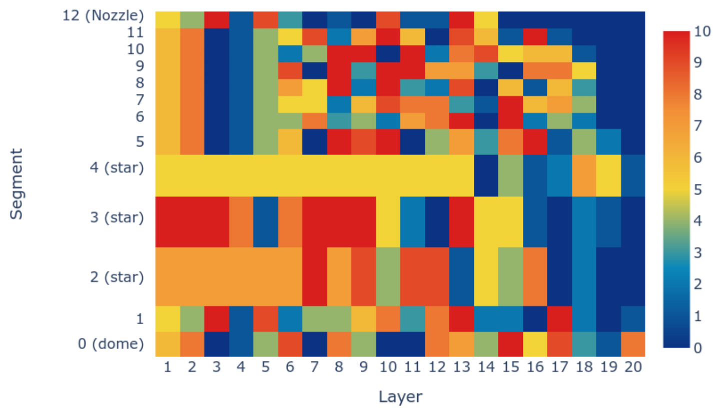
(a) Star shapes in segments 2-4.



(b) Propellant distribution in the cylindrical region.

Figure 4.13 Star shapes and cylindrical region propellant distribution for the rocket design with minimum thrust error in one run of NSGA-II/IR2 for scenario C_4. Blue regions in the star shape figures represent the propellant. The green circular rings represent the cylindrical region. In the cylindrical propellant distribution figures, blue represents the slowest-burning propellant and red represents the fastest burning propellant.

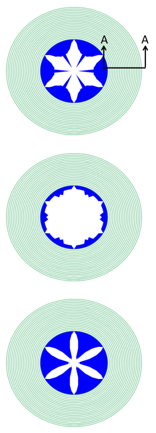


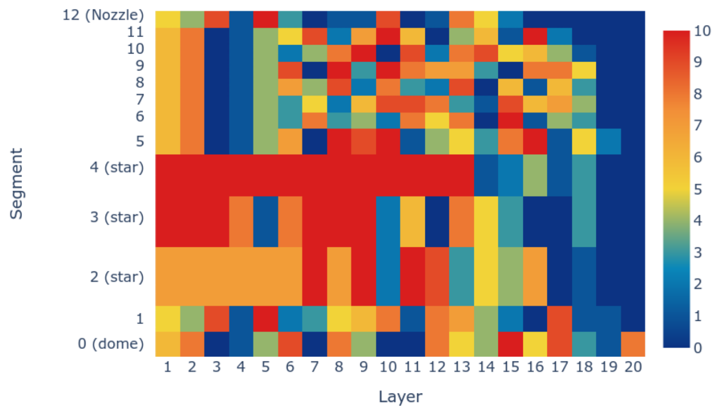


(a) Star shapes in segments 2-4.

(b) Propellant distribution in the cylindrical region.

Figure 4.14 Star shapes and cylindrical region propellant distribution for the rocket design with minimum residue in one run of NSGA-II/IR2 for scenario C_4.





(a) Star shapes in segments 2-4.

(b) Propellant distribution in the cylindrical region.

Figure 4.15 Star shapes and cylindrical region propellant distribution for a rocket design with an intermediate thrust error and residue lying in the middle region of the Pareto Front in one run of NSGA-II/IR2 for scenario C_4.

4.7 Overall Performance of Proposed Methods

Each of the three case studies (879- and 1,479-variable truss designs and 544-variable rocket design) are ranked 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 sums of ranks of each algorithm over all 3 case studies are calculated, and the algorithms are ranked accordingly, shown in Table 4.12. The final row of Table 4.12 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 third place. NSGA-II/IR3 is ranked last, showing that an aggressive use of available and potentially imperfect 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.

4.8 Limitations of the proposed MOEA/I framework

The primary limitation of the proposed MOEA/I framework is the absence of any user interaction during the optimization. While some level of user involvement is present at the beginning of the optimization run, none is present during the run. In addition, MOEA/I is shown to only work with inequality relationships. Other versatile relations such as power laws are not compatible with such a framework. These factors limit the utility of the MOEA/I framework in practical applications. This motivates the next part of the dissertation where a truly interactive framework able to accommodate multiple relationship types is proposed.

Table 4.12 Ranking of different NSGA-IIs on three case studies.

Scenario	Base	IR1	IR2	IR3	I-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
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
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	4	3	5	2
Rank	4	3	1	5	2
Rank-Sum	11	11	3	14	6
Final Rank	**3**	**3**	**1**	**4**	**2**

CHAPTER 5

AN INTERACTIVE KNOWLEDGE-BASED EVOLUTIONARY MULTI-OBJECTIVE ALGORITHM FRAMEWORK

Figure 5.1 shows the proposed interactive knowledge-based EMO (IK-EMO) framework. The framework starts with a description of the multi-objective optimization problem, as shown in the top-left box in the figure. In addition, if any additional problem information is available, that is also specified. The penultimate step before starting the optimization is to select a suitable EMO and methods to algorithmically extract and apply any problem knowledge. The subsequent sections describe the various components in more detail.

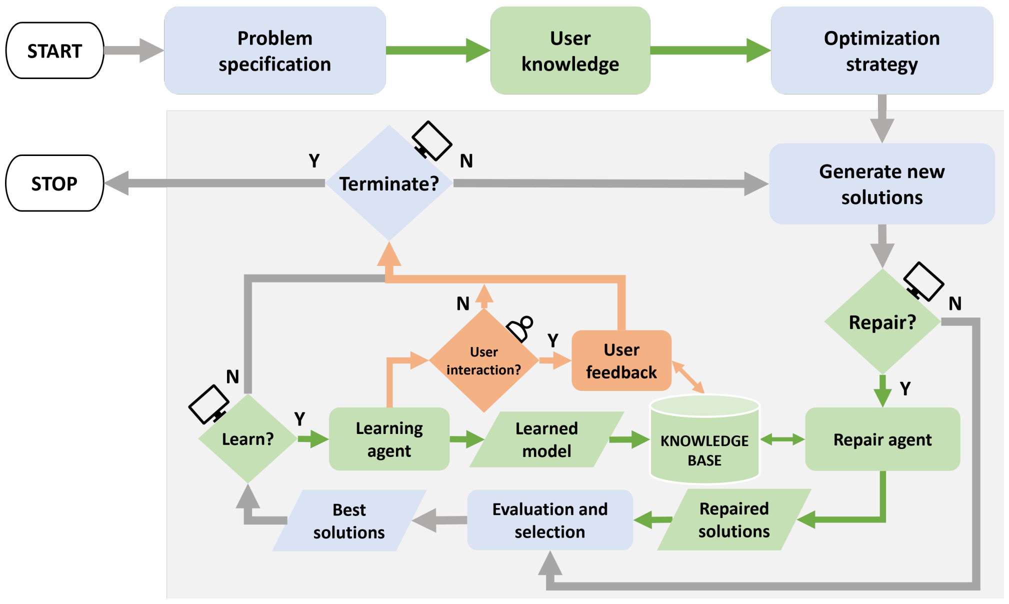


Figure 5.1 Interactive knowledge-based EMO framework (IK-EMO) showing user interaction, learning and repair agents. Blue blocks represent a normal EMO. Green blocks represent the components responsible for knowledge extraction and application. Orange blocks represent user interaction operations.

5.1 Learning agent

A learning agent is a procedure used to identify different innovization rules present among the ND solutions in a population. The rules involve a single variable or a pair of variables, as required by a rule's description. Each type of rule (inequality, power law, etc.) requires a different rule satisfaction condition. A score (within [0,1], as presented in Table 3.1) is assigned to each rule to quantify how well the rule represents the ND set. Different learning agents applicable to the rule types covered in Section 3.1 are presented below. A summary of the various rules, their scoring procedures and satisfaction conditions are provided in Table 3.1.

5.1.1 Constant rule

In order to learn constant rules, we have to analyze the values of the variable under consideration for every ND solution and check if one or more of them converge to specific values. Since variables can be of different scales and units, we need a generalized criterion to determine if a variable is taking on a constant value. First, the median ($\tilde{x}_i$) is calculated. The proportion of ND solutions which satisfy $|x_i - \tilde{x}_i| \leq \rho_i$ is said to be the score (s_{ϕ_i}) of the constant rule $x_i = \kappa_i = \tilde{x}_i$. ρ_i is a small tolerance used for determining whether variable x_i's value is in the neighborhood of $\tilde{x}_i$. It must be defined separately for each variable. An alternative option is to normalize the variables and define a singular ρ for the normalized variable space.

To check whether a new solution ($\mathbf{x}_r$) follows $x_{ir} = \kappa_i$, we check whether x_{ir} lies in the neighborhood of κ_i using the condition: $|x_{ir} - \kappa_i| \leq \rho_i$.

5.1.2 Power law rule

In order to learn power laws ($x_i x_j^b = c$) we use the method proposed in [72] with a modification. Each variable is initially normalized to [1, 2]. A training dataset is created from the ND solution set with the logarithms of normalized variables $\hat{x}_i$ and $\hat{x}_j$ as features, leading to Eqn. 5.2:

$$\hat{x}_i \hat{x}_j^b = c, \tag{5.1}$$

$$\Rightarrow \quad \log \hat{x}_i = \beta \log \hat{x}_j + \epsilon, \tag{5.2}$$

where $\beta = -b$ is the weight and $\epsilon = \log c$ is the intercept. Normalization prevents 0 or negative values from appearing in the logarithm terms. Then we apply ordinary least squares linear regression to the logarithm of $\hat{x}_i$ and $\hat{x}_j$. Linear regression finds the best-fit line for the training data defined by the parameters β and ϵ. In order to evaluate the quality of the fit, we use the coefficient of determination (R^2) metric. A new solution ($\mathbf{x}_r$) follows the power law given in Equation 5.1 if the difference between the actual value (x_{ir} or x_{jr}) and the predicted value ($\hat{x}_{ir}$ or $\hat{x}_{jr}$) is lower than a pre-defined threshold error ($e_{ij}^{\min}$). Table 3.1 shows the formulation for the satisfaction condition.

5.1.3 Equality rule

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

5.1.4 Inequality rule

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

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

5.2 Repair agent

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

5.2.1 Constant rule

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

5.2.2 Power law rule

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

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

5.2.3 Inequality and equality rules

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

$$x_j^{(k)} = x_i^{(k)} + \nu_{r1}(x_i^U - x_i^{(k)}), \qquad \text{for } x_i^{(k)} \le x_j^{(k)}, \tag{5.3}$$

$$x_j^{(k)} = \frac{x_i^{(k)} - \nu_{r2}x_i^U}{1 - \nu_{r2}}, \qquad \text{for } x_i^{(k)} \ge x_j^{(k)}. \tag{5.4}$$

Three different rule adherence (RA) schemes are considered. IQ-RA1 uses $\nu_{r1} = \mu_{\nu 1}$ and $\nu_{r2} = \mu_{\nu 2}$ (tight adherence with no standard deviation), which are computed as the means of ν_1 and ν_2 from ND solutions during the learning process, as follows:

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

For IQ-RA2, $\nu_{r1} \in \mathcal{N}(\mu_{\nu 1}, \sigma_{\nu 1})$ and $\nu_{r2} \in \mathcal{N}(\mu_{\nu 2}, \sigma_{\nu 2})$ (medium adherence with one standard deviation) are used, where σ_{ν_1} and σ_{ν_2} are standard deviations of ν_1 and ν_2, respectively. Both

v_{r1} and v_{r2} are set to zero, if they come out to be negative. For IQ-RA3, $v_{r1}, v_{r2} \in U(0,1)$ (loose adherence with a uniform distribution) are used.

5.3 Ensemble repair agent

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

$$p_r^i(t+1) = \max\left(p_{\min},\ \alpha \frac{r_s^i}{\sum_i r_s^i} + (1-\alpha)\widehat{p_r^i}(t)\right), \tag{5.5}$$

$$\widehat{p_r^i}(t+1) = \frac{p_r^i(t+1)}{\sum\limits_i p_r^i(t+1)}, \tag{5.6}$$

where α is the learning rate, $r_s^i = \frac{n_s^i}{n_{\text{off}}}$, where n_s^i and n_{off} are the number of offspring created by the i-th operator that survive in generation t and the total number of offspring that survive in generation t, respectively. It is possible that at any point during the optimization, no solution generated by one of the repair operators survives. This might cause the corresponding selection probability to go down to zero without any possibility of recovery. To prevent this, in Equation 5.5, the probability update step ensures that a minimum selection probability ($p_{\min}$) is always assigned to each repair operator present in the ensemble. Equation 5.6 normalizes the probability values for each operator so that their total sum is one.

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

5.3.1 Mixed rule repair agent

A mixed rule repair agent is designed to work on two or more different types of rules. Since multiple rules (for example, an inequality rule and a power law rule) can show up for the same variable pair, a rule hierarchy needs to exist as defined in Section 3.2.2. Table 5.1 shows the rule hierarchical rank used for all the repair agents in this study.

Table 5.1 Rule hierarchy by rank for each repair agent.

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

5.4 User's ranking of rules

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

- Rule ranking: The user may provide a ranking of rules (rank 1 is most preferred) provided by the algorithm. The algorithm will then try to implement the rules in the rank order provided by the user.

- Rule exclusion: The user may select to remove certain rules provided by the algorithm, based on their knowledge of the problem.

- Rule specificity: The user may specify details for considering a rule further. For example, the user may specify that only variables having a correlation above a specified value should be considered. Another criterion could be to select all rules having a score greater than a threshold as rank 1 and exclude the others.

In this study, the proposed rule usage schemes (RU1-RU4) can also be considered as artificial users [56] who select a certain percentage of the learned rules every few generations. This systematically illustrates the interactive ability of IK-EMO while showing the effect of different numbers of rules used for repair on the performance.

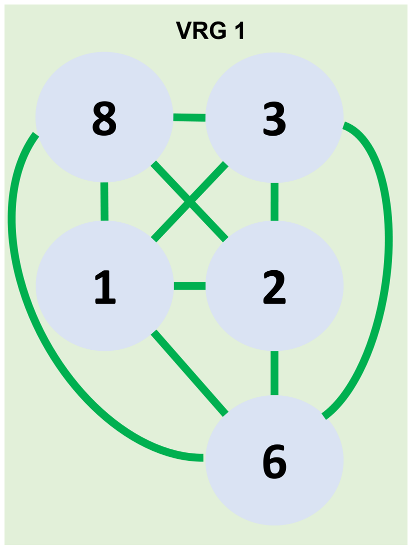
5.5 Variable relation graph (VRG)

The possible number of pair-wise relations among n variables is $\frac{n(n-1)}{2}$ or $O(n^2)$. Thus, for a large number of variables, the amount of bookkeeping required to track individual pairwise relations is large. Moreover, the observed relationships should not contradict each other. For example, for inequality rules $x_i \leq x_j$ and $x_j \leq x_k$, the transitive property can be maintained by choosing to repair x_j based on x_i, followed by repairing x_k based on x_j using Equation 5.3. But repairing both x_j and x_k separately based on x_i can potentially contradict the rule $x_j \leq x_k$. To solve these two challenges, we propose using a graph-based data structure, called variable relation graph (VRG), to encode and track relationships observed between multiple variable pairs. A customized graph-traversal algorithm ensures that all repairs are performed with minimal or no contradictions. In the following sections, steps 1 to 5 show the process of using learning agents to construct a VRG (learning phase). A learning interval (T_L) is defined as the number of generations or function evaluations (FEs) after which a new learning phase begins. Step 6 shows the process of applying the VRG to repair an offspring solution using one or more repair agents (repair phase). A repair interval (T_R) is defined as the number of generations or FEs between any two repair phases.

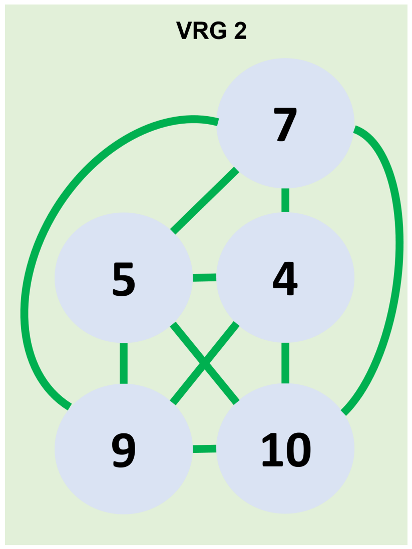
5.5.1 Create a complete VRG

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

Figure 5.2.



(a) Group G_1.

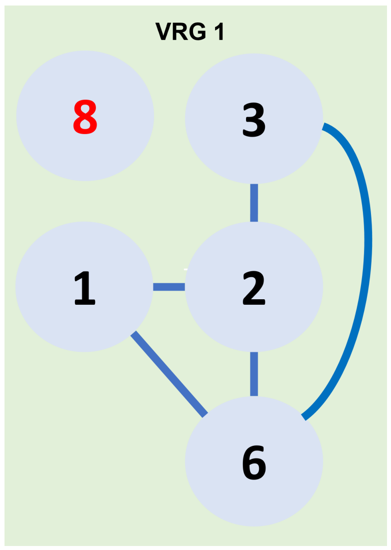


(b) Group G_2.

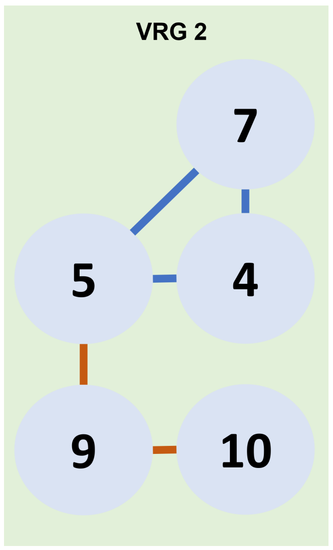
Figure 5.2 Ten variables in two non-interacting groups are represented in complete graphs.

5.5.2 Rule selection

In this step, learned rules are used to modify the VRGs according to two selection criteria. First, all rules having a score (defined in Table 3.1) above a certain threshold (s_{min}) are considered. Second, they are applied in the order of user's preference ranking. A connection may be removed if it does not satisfy the selection criteria. If a single-variable (constant) rule satisfies the selection criterion, then the corresponding node is removed from the VRG and that rule will be implemented separately. If no two-variable rule involving x_i and x_j satisfies the minimum score criterion, the corresponding VRG edge (i-j) is removed. An example is shown in Figure 5.3, which uses the rule hierarchy for mixed rule repair operators (third row) shown in Table 5.1, except that inequalities are ranked 3 for illustration here. A blue or brown edge represents a power law rule or an inequality rule, respectively. An edge ranking is also assigned based on the rule hierarchy. In this case, edges representing power laws and inequalities will be ranked 1 and 2 by default, unless overruled by the user. Both graphs have a reduced number of edges after the rule selection process is complete. Node 8 in Figure 5.3a (marked in red) is found to have a constant rule associated with it and hence removed. In Figure 5.3b, variables (x_5, x_9) and (x_9, x_{10}) are not related by power laws having a score greater than $s_{\min}$. However, they are found to follow inequality relationships with a score greater than $s_{\min}$. Hence, they are connected by brown edges. The rest of the edges represent power law rules and are marked by blue. The approach to set the direction of the edges is discussed next.



(a) Group G_1.



(b) Group G_2.

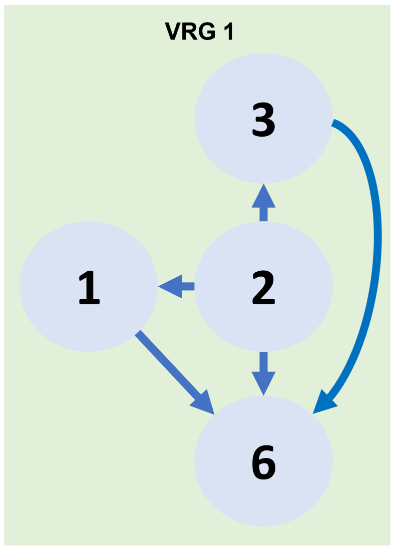
Figure 5.3 Rule selection.

5.5.3 Create a directed acyclic VRG

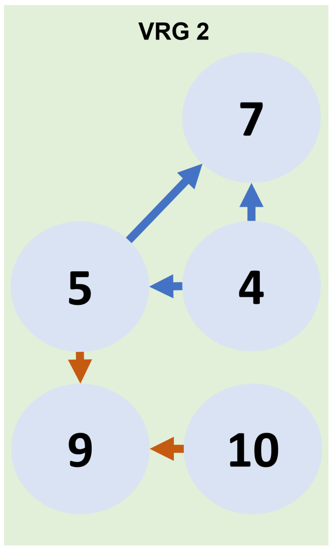
In order to apply a repair agent to the VRG, it needs to be converted to a directed acyclic graph (DAG). This step ensures graph traversal is possible without getting stuck in loops. The members of every group G_k are randomly permuted to create a sequence D_k. If i appears before j in D_k, an undirected edge between nodes i and j is converted to a directed edge from i to j. In the example shown in Figure 5.4, two random sequences $D_1 = (2, 1, 3, 6)$ and $D_2 = (10, 4, 5, 9, 7)$ are created for groups G_1 and G_2, respectively. Since node 2 appears before node 1 in D_1, a blue arrow goes from node 2 to node 1, as shown in the figure. This process is repeated for every population member so as to create diverse VRGs.

5.5.4 Transitive reduction

Next, a transitive reduction [110] is performed on the VRG corresponding to each variable group. For VRGs having both power law and inequality edges, transitive reduction is performed on subgraphs consisting only of the edges of the same type. This step eliminates redundant directed



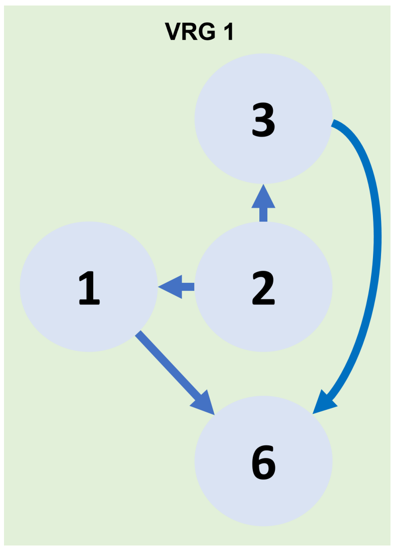
(a) Group G_1.



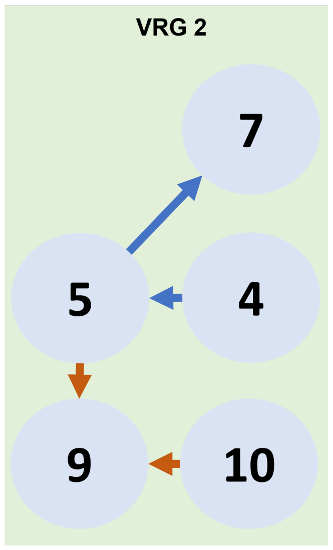
(b) Group G_2.

Figure 5.4 Creating a directed acyclic VRG.

edges between two different rule types. An example of eliminating an arrow from node 2 to node 6 is shown in Figure 5.5a.



(a) Group G_1.

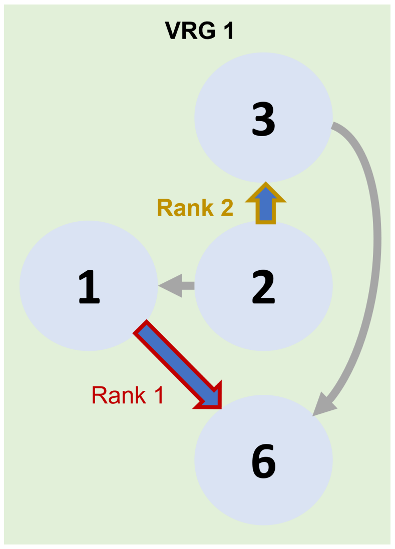


(b) Group G_2.

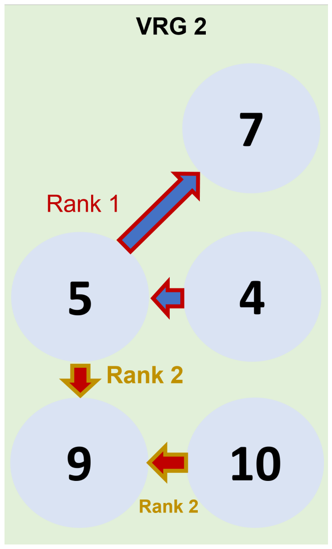
Figure 5.5 Transitive reduction.

5.5.5 Modify VRG according to user feedback

A user can provide feedback in the form of a ranking, or select only a subset of the available rules. In the former case, the VRG edge rankings are updated to reflect the user's choice. Edges corresponding to the rules discarded by the user are removed. Figure 5.6a shows an example where the rule involving x_1 and x_6 are ranked 1 (marked by arrows with a red border) and x_2 and x_3 are ranked 2 (marked by arrows with a dark yellow border). The gray edges represent the rules discarded by the user. Figure 5.6b shows a similar ranking process.



(a) Group G_1.



(b) Group G_2.

Figure 5.6 Implementing user feedback.

5.5.6 Repair new offspring solutions

For every new solution, the corresponding VRGs are traversed. A random rank 1 starting point is selected and the VRG is traversed recursively in a depth-first fashion. From every node, the algorithm first moves forward via the outgoing edges and repairs the connected node based on the current node. Once all outgoing edges are traversed, and the algorithm comes back to the same node, traversal is performed by following the incoming edges. This is repeated for all ranks. Algorithm 5.1 presents the pseudocode of the repair process. For ease of understanding, some of the terminology used in the pseudocode is explained in this section. In the pseudocode, the VRG data structure has the attributes Nodes and Edges. The Edges attribute representing an edge (i, j) has multiple sub-attributes: StartVertex (i in this case), EndVertex (j in this case), EdgeType (rule type and correspdonding repair agent), EdgeRank (rank of an edge). A function *TraverseGraph* is used which recursively traverses the VRG from a random start node for a particular rule rank. The function *Repair* called by *TraverseGraph* calls the correct repair agent based on EdgeType.

Algorithm 5.1 VRG traversal and repair pseudocode.

```
Require: New solution set ($\mathcal{X}_r$), variable groups ($G$), VRGs for every solution and group, rule hierarchy.
Ensure: Repaired solution set $\mathcal{X}_r$.
 1: function TraverseGraph(x, Graph, CurrentNode, PreviousNode, NodesVisited, CurrentRank)
 2:     if CurrentNode in NodesVisited then
 3:         return
 4:     end if
 5:     CurrentEdges ← Graph.Edges[CurrentNode];
 6:     for each outgoing edge (e) in CurrentEdges do
 7:         NextNode ← e.EndVertex;
 8:         if NextNode not in NodesVisited then
 9:             EdgeType ← e.EdgeType;
10:             EdgeRank ← e.EdgeRank;
11:             if EdgeRank = CurrentRank then
12:                 Repair(x, CurrentNode, NextNode, EdgeType, EdgeRank);
13:             end if
14:             TraverseGraph(x, Graph, NextNode, CurrentNode, NodesVisited);
15:         end if
16:     end for
17:     for each incoming edge (e) in CurrentEdges do
18:         NextNode ← e.StartVertex;
19:         if NextNode not in NodesVisited and NextNode ≠ PreviousNode then
20:             EdgeType ← e.EdgeType;
21:             EdgeRank ← e.EdgeRank;
22:             if EdgeRank = CurrentRank then
23:                 Repair(x, CurrentNode, NextNode, EdgeType, EdgeRank);
24:             end if
25:             TraverseGraph(x, Graph, NextNode, CurrentNode, NodesVisited);
26:         end if
27:     end for
28:     Add CurrentNode to NodesVisited;
29: end function
30: for each group $G_k$ in $G$ do                                  ▷ Repair procedure begins
31:     for each solution $x$ in $\mathcal{X}_r$ do
32:         CurrentGraph ← VRG assigned to x for $G_k$;
33:         for CurrentRank = 1, 2, ..., $n_{ranks}$ do
34:             StartNode ← Select random node having atleast one edge of rank CurrentRank;
35:             TraverseGraph(x, CurrentGraph, StartNode, NULL, [], CurrentRank);
36:         end for
37:     end for
38: end for
```

5.6 Simply-supported stepped beam design

5.6.1 Problem background and formulation

Beam design problems are common in the literature [6, 111] and can be used to benchmark an optimization algorithm. In this study, we consider a simply-supported stepped beam design with multiple segments having a rectangular cross-section. An example with five segments is shown in Figure 5.7. A vertical load of 2 kN is applied at the middle of the beam. All n_{seg} segments

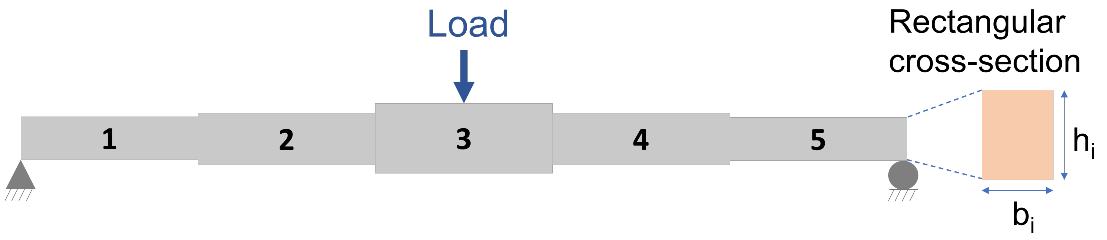


Figure 5.7 Simply-supported stepped beam with five segments.

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

$$\text{Minimize } V(\mathbf{x}) = \sum_{i=1}^{n_{\text{seg}}} b_i h_i l_i, \tag{5.7}$$

$$\text{Minimize } \Delta(\mathbf{x}) = \max_{i=1}^{n_{\text{seg}}} \delta_i(\mathbf{x}), \tag{5.8}$$

$$\text{Subject to } \max_{i=1}^{n_{\text{seg}}} \sigma_i(\mathbf{x}) \le \sigma_{\max}, \tag{5.9}$$

$$\max_{j=1}^{n_{\text{seg}}} \delta_j(\mathbf{x}) \le \delta_{\max}, \tag{5.10}$$

$$a_L \le a_i \le a_U, \text{ for } i = 1, \ldots, n_{\text{seg}}. \tag{5.11}$$

Here, two cases with 39 and 59 segments are considered. Problem parameters are described in Table 5.2.

Table 5.2 Number of decision variables and constraints for 39 and 59-segment stepped beams.

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

5.6.2 Experimental settings

NSGA-II [15], a state-of-the-art MOEA, is applied with the proposed IK-EMO procedure to solve both cases. This problem is intended to demonstrate the performance of our proposed algorithm with minimal initial user knowledge. Thus, no grouping information is provided, resulting in all variables being in a single group. IK-EMO is combined separately with each repair agent described in Section 5.2. In addition, there are two cases where mixed relationships are used: the first case with PL-RA2 and IQ-RA2, and the second case with PL-RA-E and I-ES. The rule hierarchy is described in Table 5.1.

Four rule usage schemes RU1, RU2, RU3 and RU4 select the top 10%, 20%, 50% and 100% of the learned rules sorted according to their scores. They also act as artificial users with a consistent behavior. Each rule usage scheme is paired with one or more repair agents. Eight cases with a single repair agent are considered: PL-RA1, PL-RA2, PL-RA3, PL-RA-E, IQ-RA1, IQ-RA2, IQ-RA3, IQ-RA-E. Two cases with a combination of repair agents are considered: one with PL-RA2 and IQ-RA2, and the other with PL-RA-E and IQ-RA-E. From Table 3.1, ρ_i is set to be 0.1, ε_{ij} is set as 0.1, and e_{ij}^{min} is set to 0.01. Table 5.3 shows the parameter settings for this problem.

For each combination of a repair agent and user, 20 runs are performed and the Hypervolume (HV) [104] values are recorded at the end of each generation. The Wilcoxon rank-sum test [108] is used to compare the statistical performance of the algorithms tested here with respect to the best performing algorithm for each scenario.

5.6.3 Experimental results and discussion

Tables 5.4 and 5.5 show the optimization results for the 39 and 59-segment stepped beam problems, respectively. Base NSGA-II results without any rule extraction and repair are shown in the first row.

Table 5.3 Parameter settings of IK-EMO.

Parameter	Value
Population size	Problem-specific
Maximum generations	Problem-specific
Mutation operator	Polynomial mutation [11]
Mutation probability (p_m) and index (η_m)	$1/n_{var}$, 50
Crossover operator	SBX [3]
Crossover probability (p_c) and index (η_c)	0.9, 30
Minimum rule score, $s_{\min}$	0.7
Learning interval (T_L, in generations)	10
Repair interval (T_R, in generations)	10
α and $p_{\min}$ in Equation 5.5	0.5, 0.1
Rule parameters ρ_i, ε_{ij}, $e_{ij}^{\min}$	Problem-specific
User feedback lag, T_U in Section 5.10	User-dependent

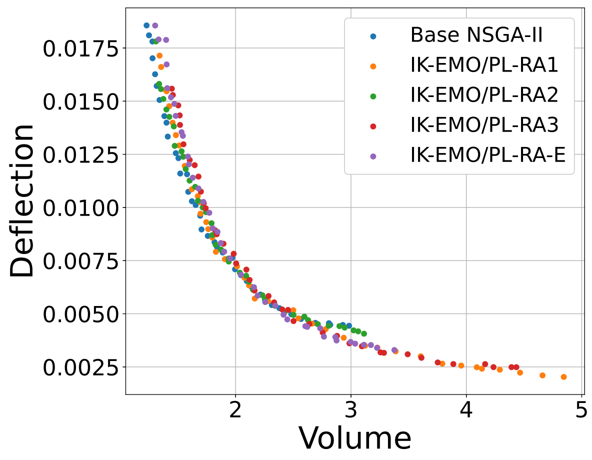
The best performance case in each row is marked in bold. For every column, the best performing algorithm is marked with a shaded gray box. The Wilcoxon p-values show the relative performance of each algorithm with the column-wise best performance. Algorithms with a statistically similar performance to the column-wise best are shown in italics. The ND front obtained in a particular run using the power law repair operators for RU2 are shown in Figures 5.8a and Figure 5.8c for the 39 and 59-segment cases, respectively. The corresponding median HV plot over the course of the optimization run are shown in Figures 5.8b and Figure 5.8d, respectively.

Table 5.4 FEs required to achieve HV^T = 0.81 for 39-segment beams. Best performing algorithm for row is marked in bold. Best performing algorithm in each column is marked by a shaded gray box. Algorithms with statistically similar performance to the best algorithm column-wise are marked in italics. The corresponding Wilcoxon p-values are given in braces.

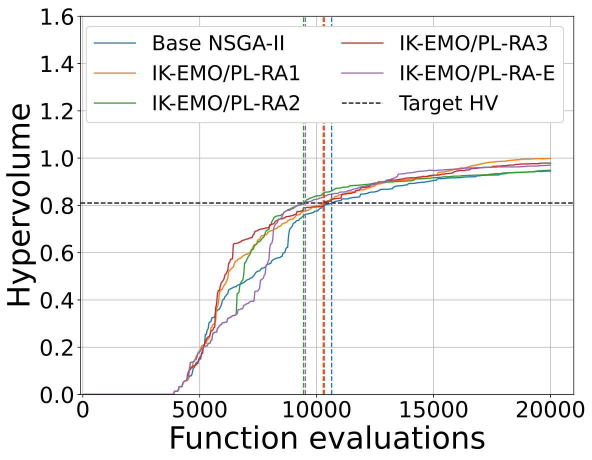
Rule Type	Repair agent	RU1	RU2	RU3	RU4
None	None (base)	10.6k ± 1.0k (p = 0.0145)	10.6k ± 1.0k (p=0.0118)	10.6k ± 1.0k (p=0.0238)	10.6k ± 1.0k (p=0.0412)
	PL-RA1	10.4k ± 0.8k (p=0.0416)	**10.3k** ± **1.0k** (p=0.0225)	10.5k ± 0.9k (p=0.0420)	10.6k ± 1.0k (p=0.0319)
Power	PL-RA2	*9.8k* ± *0.8k* (p=0.0661)	**9.4k** ± **0.9k**	9.5k ± 1.2k	*10.2k* ± *1.3k* (p=0.0551)
law rule	PL-RA3	10.4k ± 0.9k (p=0.0195)	10.3k ± 1.0k (p=0.0422)	**9.8k** ± **1.0k** (p=0.0106)	11.6k ± 1.0k (p=0.0147)
	PL-RA-E	9.6k ± 1.0k	***9.5k*** ± ***1.0k*** (p=0.0762)	*9.6k* ± *1.4k* (p=0.0841)	9.9k ± 1.2k
	IQ-RA1	10.7k ± 0.7k (p=0.0016)	**10.5k** ± **1.0k** (p=0.0471)	10.6k ± 0.9k(p=0.0483)	10.8k ± 1.0k (p=0.0308)
Inequality	IQ-RA2	**10.2k** ± **0.9k (p=0.0125)**	10.3k ± 1.1k (p=0.0263)	10.3k ± 0.8k(p=0.0486)	10.7k ± 1.2k (p=0.0210)
rule	IQ-RA3	10.7k ± 1.2k (p=0.0483)	10.6k ± 1.1k(p=0.0340)	**10.5k** ± **1.0k** (p=0.0318)	10.6k ± 1.4k (p=0.0247)
	IQ-RA-E	10.6k ± 0.8k (p=0.0463)	**10.5k** ± **0.9k** (p=0.0207)	10.6k ± 1.3k(p=0.0342)	10.6k ± 1.6k (p=0.0177)
Mixed	PL-RA2+IQ-RA2	*9.8k* ± *0.9k* (p=0.0517)	***9.6k*** ± ***1.0k*** (p=0.0957)	*9.7k* ± *1.1k* (p=0.0586)	*10.4k* ± *0.8k* (p=0.0778)
rule	PL-RA-E+IQ-RA-E	***9.7k*** ± ***1.2k*** (p=0.0913)	*9.6k* ± *0.4k* (p=0.0713)	*9.8k* ± *0.7k* (p=0.0616)	*10.0k* ± *0.7k* (p=0.0506)

Table 5.5 FEs required to achieve HV^T = 0.75 for 59-segment beams. Best performing algorithm for row is marked in bold. Best performing algorithm in each column is marked by a shaded gray box. Algorithms with statistically similar performance to the best algorithm column-wise are marked in italics. The corresponding Wilcoxon p-values are given in braces.

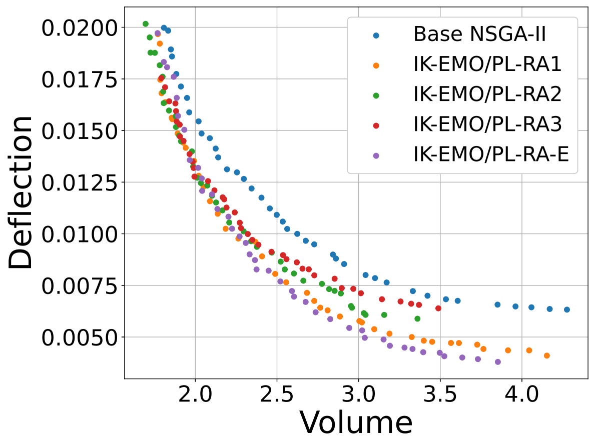
Rule Type	Repair agent	RU1	RU2	RU3	RU4
None	None (base)	19.0k ± 2.5k (p=0.0102)	19.0k ± 2.5k (p=0.0015)	19.0k ± 2.5k (p=0.0011)	19.0k ± 2.5k (p=0.0027)
Power law rule	PL-RA1	15.6k ± 1.8k (p=0.0105)	**14.1k ± 2.3k**	15.2k ± 3.1k (p=0.0164)	16.1k ± 2.5k (p=0.0371)
	PL-RA2	14.8k ± 1.8k	15.0k ± 2.0k (p=0.0225)	***14.4k ± 2.8k*** (p=0.1015)	*15.5k ± 1.8k* (p=0.0510)
	PL-RA3	16.0k ± 2.2k (p=0.0042)	**15.8k ± 1.5k** (p=0.0218)	16.6k ± 3.5k (p=0.0215)	17.0k ± 2.6k (p=0.0446)
	PL-RA-E	*14.9k ± 3.1k* (p=0.1165)	***14.2k ± 3.6k*** (p=0.0911)	14.2k ± 2.9k	14.9k ± 2.5k
Inequality rule	IQ-RA1	16.6k ± 4.1k (p=0.0215)	**16.0k ± 2.6k** (p=0.0411)	16.6k ± 2.0k (p=0.0182)	18.1k ± 2.2k (p=0.0341)
	IQ-RA2	16.8k ± 4.0k (p=0.0193)	**15.9k ± 2.9k** (p=0.0365)	16.8k ± 3.0k (p=0.0179)	17.6k ± 2.6k(p=0.0335)
	IQ-RA3	**16.5k ± 4.3k** (p=0.0317)	17.2k ± 3.1k (p=0.0357)	17.2k ± 3.2k (p=0.0155)	17.9k ± 2.4k(p=0.0273)
	IQ-RA-E	16.6k ± 3.5k (p=0.0287)	**16.4k ± 4.2k** (p=0.0282)	16.9k ± 2.7k (p=0.0293)	17.8k ± 1.9k(p=0.0228)
Mixed rule	PL-RA2+IQ-RA2	*15.1k ± 3.0k* (p=0.0583)	***14.5k ± 3.1k*** (p=0.0917)	*14.6k ± 3.0k* (p=0.0715)	*15.4k ± 2.5k* (p=0.0713)
	PL-RA-E+IQ-RA-E	*14.9k ± 2.6k* (p=0.0917)	***14.4k ± 2.9k*** (p=0.0663)	*14.6k ± 2.6k* (p=0.0681)	*15.2k ± 2.2k* (p=0.0558)



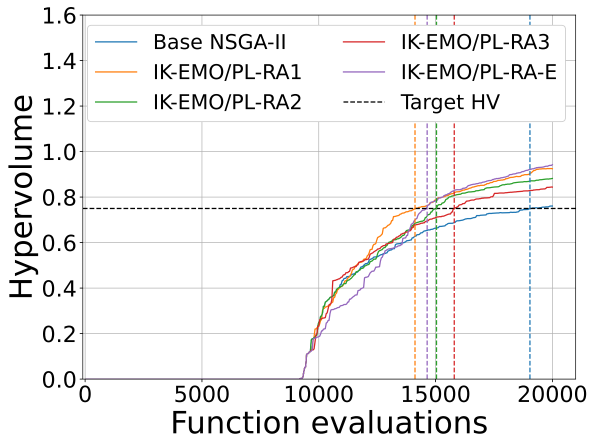
(a) Pareto Front for one run.



(b) HV plot over 20 runs.



(c) ND Front for one run.



(d) HV plot over 20 runs.

Figure 5.8 ND fronts and hypervolume plots obtained by IK-EMO with RU2 and power law repair agents for: (a-b) 39-segment, and (c-d) 59-segment stepped beam problem.

The results show many interesting observations as stated below.

5.6.3.1 General observations

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

5.6.3.2 Power law vs inequality rules

As can be seen from the table, for both 39- and 59-segment problems, the power law repair alone results in the best performance for each user case. Inequality rule-based repair alone in general results in worse performance compared to power-law-based repair. One possible reason could be the greater versatility of power laws in modeling complex relationships compared to simple inequality rules.

5.6.3.3 Best performing algorithm for each rule usage scheme

In the 39-segment case, PL-RA-E is the best performer for both RU1 and RU4. For RU2 and RU3, PL-RA2 is the best performer. It is to be noted that PL-RA-E has statistically similar performance to PL-RA2 for both RU2 and RU3. This shows that the ensemble method can be used to get good performance without the need for selecting a proper repair process. For the 59-segment case, PL-RA2 and PL-RA1 are the best performers for RU1 and RU2. In the cases of RU3 and RU4, PL-RA-E gives the best performance, with PL-RA2 having a comparable performance. PL-RA1 and PL-RA3 do not show comparable performance with PL-RA2 or PL-RA-E in most cases. For PL-RA1, adhering closely to the learned power law rules constrains NSGA-II in finding good solutions. PL-RA3 introduces a large amount of variance which is detrimental to the optimization process. A compromise between these two extremes, provided by PL-RA2 or PL-RA-E, is the logical step.

Figure 5.8 illustrates the results when RU2 is combined with the power law repair operators for both problem cases. The difference in the quality of solutions obtained after 20,000 FEs is prominent in the 59-segment ND front. In the median HV plots it is seen that the FEs required to reach HV^T for PL-RA-E is close to the number needed by the best performing repair agent. Base NSGA-II and the repair operators all give good quality solutions at the end of the run for the 39-segment case. However, for the 59-segment case, base NSGA-II performs significantly worse.

5.6.3.4 Relative performance of each rule usage scheme

It can be seen from Tables 5.4 and 5.5 that RU2 produces the best performance in 6 out of 10 cases for the 39-segment case, and 8 out of 10 cases for the 59-segment case. This shows that in terms of rule usage, using too few or too many of the learned rules is not effective in improving the algorithm's performance.

5.6.3.5 Mixed relation repair agents

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

5.7 Optimal Power Flow Problem

5.7.1 Problem background and formulation

Optimal power flow (OPF) is a common problem in power system engineering with MOEAs being used to solve the problem [112, 113, 113, 114, 115, 116, 117, 118]. The following objective functions are minimized: fuel cost, emissions, voltage deviation, and real power loss. In many cases, one or more of these objectives are considered in the literature, with the rest being kept as constraints. In this study, we consider two objectives: minimizing fuel cost and reducing fossil fuel emissions. Voltage deviation and power loss are kept as constraints. This version of the OPF problem is also known as the environmental economic dispatch (EED) problem [113].

$$\text{Minimize } C_F(\mathbf{P_G}, \mathbf{V_G}) = \sum_{i=1}^{N_G} \left(a_i + b_i P_{Gi} + c_i P_{Gi}^2 \right), \tag{5.12}$$

$$\text{Minimize } C_E(\mathbf{P_G}, \mathbf{V_G}) = \sum_{i=1}^{N_G} \left(\alpha_i + \beta_i P_{Gi} + \gamma_i P_{Gi}^2 + \zeta_i e^{(\lambda_i P_{Gi})} \right), \tag{5.13}$$

$$\text{Subject to } \sum_{i=1}^{N_{bus}} (P_i - P_D - P_L) = 0, \tag{5.14}$$

$$VD_{\min} \leq VD \leq VD_{\max}, \quad P_{Lmin} \leq P_L \leq P_{Lmax},$$

$$Q_{Gimin} \leq Q_{Gi} \leq Q_{Gimax}, \quad P_{smin} \leq P_s \leq P_{smax},$$

$$V_{smin} \leq V_s \leq V_{smax}, \quad V_{PQimin} \leq V_{PQi} \leq V_{PQimax},$$

where C_F is the fuel cost, C_E is the emission cost, N_G is the number of generators, P_{Gi} is the real power output and V_{Gi} is the voltage output of the i^{th} generator, (a_i, b_i, c_i) are the fuel cost coefficients, $(\alpha_i, \beta_i, \gamma_i, \zeta_i, \lambda_i)$ are the emission cost coefficients. VD is the total voltage deviation of all the load buses, P_L is the total real power loss, Q_{Gi} is the reactive power output of the i^{th} generator, P_s is the real power output and V_s is the voltage output of the slack bus, V_{PQi} is the voltage at the i^{th} load/P-Q bus. P_D is the power demand and N_{bus} is the total number of buses. A load flow analysis must be performed to satisfy the equality constraint. We use MATPOWER [119] as the load flow solver. We consider IEEE 118-bus and 300-bus systems in this study.

5.7.2 Experimental settings

The bus details, along with the numbers of decision variables and constraints, are given in Table 5.6. The types of decision variables and their corresponding ranges are given in Table 5.7.

Table 5.6 IEEE bus system specifications.

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

Table 5.7 OPF decision variable types and ranges.

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

Experimental settings are the same as in the stepped beam problem except that the population size is set to be 50 and the maximum number of generations is set as 400 for both IEEE 118 and 300-bus systems. From Table 3.1, ρ_i and ε_{ij} are set as 1, and e_{ij}^{min} is set to 0.01. Two variable groups are defined and shown in Table 5.8.

5.7.3 Experimental results and discussion

Tables 5.9 and 5.10 show the optimization results for the IEEE 118- and 300-bus systems, respectively. The ND fronts obtained in a single run using four power law repair methods for RU2 are

Table 5.8 OPF variable groups.

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

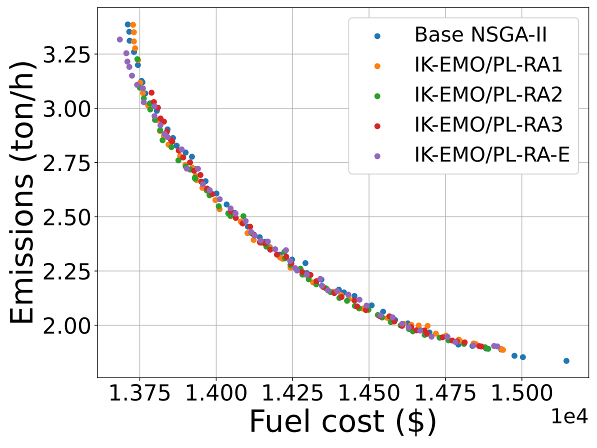
shown in 5.9a and Figure 5.9c for the IEEE 118 and 300-bus systems, respectively. The corresponding median HV plots over the course of the optimization are shown in Figures 5.9b and Figure 5.9d, respectively.

Table 5.9 FEs required to achieve HV^T = 0.74 for IEEE 118-bus system. Best performing algorithm for row is marked in bold. Best performing algorithm in each column is marked by a shaded gray box. Algorithms with statistically similar performance to the best algorithm column-wise are marked in italics. The corresponding Wilcoxon p-values are given in braces.

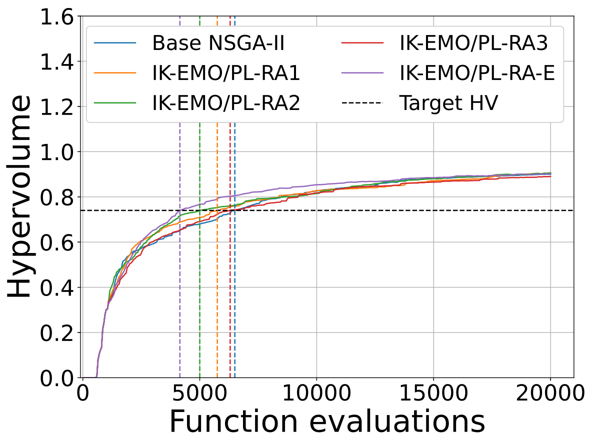
Rule Type	Repair agent	RU1	RU2	RU3	RU4
None	None (base)	6.5k ± 0.6k (p=0.0216)	6.5k ± 0.6k (p=0.0286)	6.5k ± 0.6k (p=0.0337)	6.5k ± 0.6k (p=0.0432)
Power law rule	PL-RA1	5.9k ± 0.2k (p=0.0101)	**5.8k** ± **0.4k** (p=0.0417)	5.8k ± 0.5k (p=0.0119)	6.0k ± 0.2k
	PL-RA2	4.5k ± 0.4k	***5.0k*** ± ***0.9k*** (p=0.0805)	5.2k ± 0.2k	*6.1k* ± *0.3k* (p=0.0813)
	PL-RA3	6.8k ± 0.5k (p=0.0152)	6.3k ± 0.7k (p=0.0398)	**6.2k** ± **0.4k** (p=0.0817)	7.0k ± 0.6k (p=0.0086)
	PL-RA-E	*4.7k* ± *0.4k* (p=0.0656)	**4.2k** ± **0.5k**	*5.4k* ± *0.3k* (p=0.0680)	*6.1k* ± *0.2k* (p=0.0727)
Inequality rule	IQ-RA1	6.6k ± 0.4k (p=0.00119)	6.9k ± 0.7k (p=0.0255)	**6.1k** ± **0.1k** (p=0.0341)	7.0k ± 0.5k (p=0.0338)
	IQ-RA2	6.6k ± 0.2k (p=0.0065)	6.8k ± 0.5k (p=0.0021)	**6.4k** ± **0.3k** (p=0.0279)	6.8k ± 0.1k (p=0.0332)
	IQ-RA3	6.5k ± 0.4k (p=0.0138)	6.4k ± 0.2k (p=0.0018)	**6.1k** ± **0.3k** (p=0.0275)	7.5k ± 0.4k(p=0.0320)
	IQ-RA-E	6.5k ± 0.1k (p=0.0129)	**6.3k** ± **0.4k** (p=0.0121)	6.8k ± 0.3k (p=0.0116)	7.0k ± 0.2k (p=0.0112)
Mixed rule	PL-RA2 + IQ-RA2	***4.8k*** ± ***0.3k*** (p=0.0722)	***4.8k*** ± ***0.4k*** (p=0.0713)	*5.4k* ± *0.3k* (p=0.0841)	*6.2k* ± *0.1k* (p=0.0748)
	PL-RA-E + IQ-RA-E	*4.6k* ± *0.1k* (p=0.0903)	***4.4k*** ± ***0.2k*** (p=0.0667)	5.6k ± 0.1k (p=0.0144)	*6.2k* ± *0.2k* (p=0.0919)

Table 5.10 FEs required to achieve HV^T = 0.70 for IEEE 300-bus system. Best performing algorithm for row is marked in bold. Best performing algorithm in each column is marked by a shaded gray box. Algorithms with statistically similar performance to the best algorithm column-wise are marked in italics. The corresponding Wilcoxon p-values are given in braces.

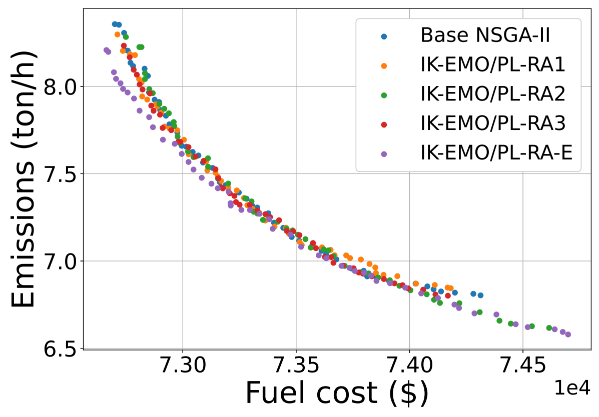
Rule Type	Repair agent	RU1	RU2	RU3	RU4
None	None (base)	17.5k ± 1.2k (p=0.0142)	17.5k ± 1.2k (p=0.0205)	17.5k ± 1.2k (p=0.0130)	17.5k ± 1.2k (p=0.0091)
Power law rule	PL-RA1	***14.8k ± 0.7k*** (p=0.0878)	15.9k ± 0.8k (p=0.0110)	16.2k ± 0.6k (p=0.0035)	16.8k ± 0.6k (p=0.0063)
	PL-RA2	*15.1k ± 0.6k* (p=0.0753)	***14.9k ± 0.5k*** (p=0.0518)	15.9k ± 0.2k (p=0.0239)	*14.9k ± 0.2k* (p=0.0657)
	PL-RA3	18.5k ± 0.3k (p=0.0413)	**17.5k ± 1.0k** (p=0.0017)	19.3k ± 1.2k (p=0.0181)	18.8k ± 0.5k (p=0.0025)
	PL-RA-E	14.5k ± 0.2k	**14.4k ± 0.3k**	14.8k ± 0.6k	15.6k ± 0.9k
Inequality rule	IQ-RA1	**15.2k ± 0.9k** (p=0.0315)	16.0k ± 1.1k (p=0.0033)	18.5 ± 1.3k (p=0.0059)	18.2 ± 1.0k (p=0.0024)
	IQ-RA2	16.9k ± 1.2k (p=0.0122)	**16.5k ± 1.0k** (p=0.0015)	17.6 ± 0.7k (p=0.0073)	18.0 ± 1.3k (p=0.0022)
	IQ-RA3	16.8k ± 1.0k (p=0.0286)	**16.1k ± 0.8k** (p=0.0112)	16.6 ± 1.7k (p=0.0012)	19.5 ± 1.5k (p=0.0016)
	IQ-RA-E	15.9k ± 0.8k (p=0.0252)	16.8k ± 0.8k (p=0.0104)	**15.7 ± 0.6k** (p=0.0076)	18.3 ± 1.1k (p=0.0032)
Mixed rule	PL-RA2 + IQ-RA2	*14.9k ± 0.8k* (p=0.0991)	***14.8k ± 0.7k*** (p=0.0836)	16.3k ± 0.2k (p=0.0103)	*16.0k ± 3.0k* (p=0.0528)
	PL-RA-E + IQ-RA-E	*14.7k ± 0.7k* (p=0.1013)	***14.6k ± 0.9k*** (p=0.0811)	*15.0k ± 0.8k* (p=0.0713)	*16.2k ± 0.9k* (p=0.661)



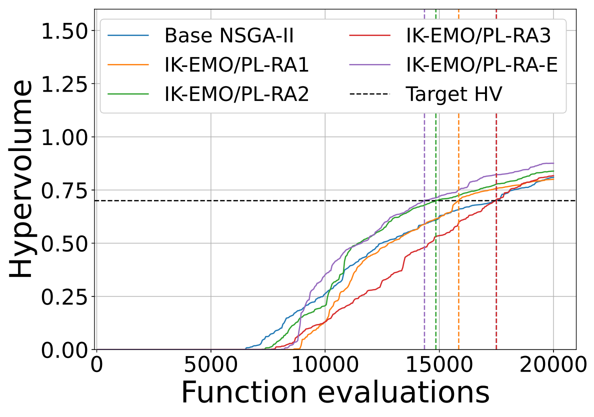
(a) ND Front for one run.



(b) HV plot over 20 runs.



(c) ND Front for one run.



(d) HV plot over 20 runs.

Figure 5.9 ND front and hypervolume plots obtained with RU2 and power law repair agents for: (a-b) IEEE 118 and (c-d) 300-bus OPF problems.

5.7.3.1 General observations

Base NSGA-II is outperformed by PL-RA1, PL-RA2 and PL-RA-E as well as the mixed rule repair agents. However, for PL-RA3 and the inequality repair operators, base NSGA-II produces comparable performance in most cases. Good performance with low population size is obtainable by the power-law-based repair agents.

5.7.3.2 Power law vs inequality rules

For both problem cases, a power law repair operator is the best performer for each user, as in the stepped beam problem. Inequality-rule-based repair operators in general result in worse performance compared to power-law-based repair operators as well as the base NSGA-II. This, as in the stepped beam problem, is a result of the power laws being able to more accurately model the inter-variable relationships. For PL-RA3, having a high variance ($2\sigma_c$) during repair is harmful to the optimization, resulting in comparable or worse performance than base NSGA-II in general.

5.7.3.3 Best performing algorithm for each user

In the IEEE 118-bus system, PL-RA-E is the best performer for RU2, with PL-RA2 showing comparable performance. For RU1 and RU3, PL-RA2 is the best performer, with PL-RA-E showing comparable performance. For RU4, PL-RA1 is the best, with PL-RA2 and PL-RA-E giving statistically similar performance. As in the case of the stepped beam problems, PL-RA-E is either the best or gives statistically similar performance. Thus, good performance can be obtained without the need to determine which power-law-based repair operator is the best.

Figure 5.9 illustrates the results with RU2 combined with the power law repair operators for both problem cases. The difference in the quality of solutions obtained after 20,000 FEs is more prominent in the IEEE 300-bus case. In the median HV plots it is seen that the number of FEs required to reach HV^T for PL-RA-E is close to that of PL-RA2. Base NSGA-II and PL-RA3 show worse performance than the others.

5.7.3.4 Relative performance of each user

It can be seen from Tables 5.9 and 5.10 that RU2 produces the best performance in 6 out of 10 cases for the IEEE 118-bus case, and 7 out of 10 cases for the IEEE 300-bus case. This shows that in terms of rule usage, using too few or too many of the learned rules is detrimental to the optimization performance in general, which is similar to the conclusions made in the stepped beam design problems.

5.7.3.5 Mixed relation repair agents

The mixed relation repair agents (PL-RA2+IQ-RA2) and (PL-RA-E+IQ-RA-E), have statistically similar performance to the best algorithm for each user and both problem cases. As in the stepped beam problems, the worse performance of the inequality repair operators does not hinder the performance of the mixed relation operators.

5.8 Truss Design Problem

This problem uses the same problem formulation given in section 4.5.

5.8.1 Experimental settings

Experimental settings are similar to those of the previous problems. Population size is set to 100 and the maximum number of generations is set as 10,000. Thus, the total computational budget comes out to be 1 million FEs. From Table 3.1, ρ_i and ε_{ij} are set as 0.1, and e_{ij}^{min} is set to 0.01. Multiple variable groups are defined for this problem based on the relative location and alignment of the beams as shown in Table 5.11.

Table 5.11 Variable groups for the 1,100-member truss cases. Each group has comparable variables having identical units and scales.

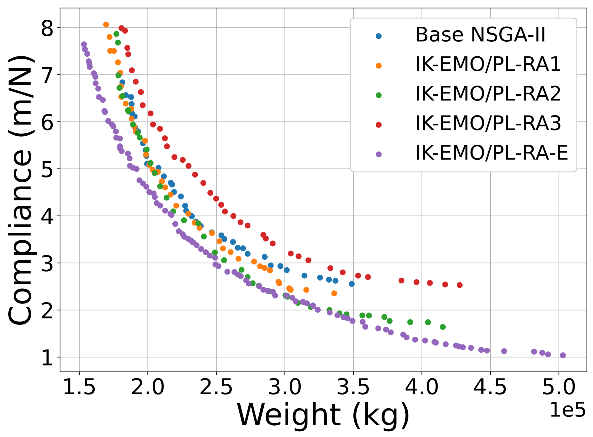
Group	Variable Type	Variable Indices
G_{t1}	l_i of vertical members	[1101 – 1179]
G_{t2}	r_i of top longitudinal members	[79 – 156], [235 – 312]
G_{t3}	r_i of bottom longitudinal members	[1 – 78], [157 – 234]
G_{t4}	r_i of vertical members	[313 – 391]

5.8.2 Experimental results and discussion

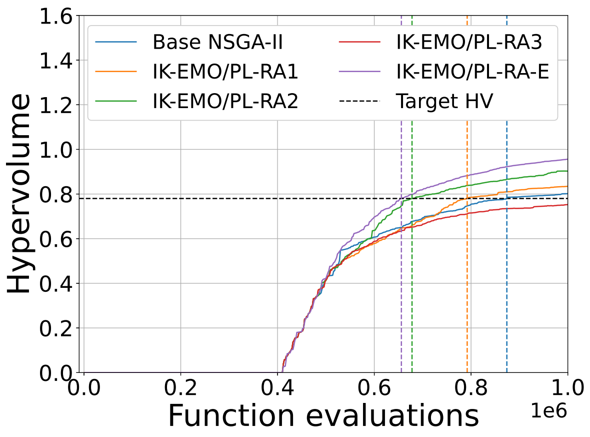
Results on the 1100-member truss problem is given in Table 5.12. The Pareto Fronts obtained in 1 run using the power law repair operators for RU2 are shown in Figure 5.10a. The corresponding median HV plots over the course of the optimization are shown in Figure 5.10b.

Table 5.12 FEs required to achieve HV^T = 0.78 for 1100-member truss. Best performing algorithm for row is marked in bold. Best performing algorithm in each column is marked by a shaded gray box. Algorithms with statistically similar performance to the best algorithm column-wise are marked in italics. The corresponding Wilcoxon p-values are given in braces.

Rule Type		RU1	RU2	RU3	RU4
None	Base	874k ± 10k (p = 0.0043)	874k ± 10k (p = 0.0017)	874k ± 10k (p = 0.0062)	874k ± 10k (p = 0.0053)
Power law	PL-RA1	802k ± 8k (p=0.0057)	792k ± 8k (p=0.0129)	**786k ± 13k** (p=0.0140)	812k ± 6k (p=0.0023)
	PL-RA2	*680k ± 11k* (p=0.0633)	***678k ± 15k*** (p=0.0793)	688k ± 10k	744k ± 17k
	PL-RA3	**963k ± 21k** (p=0.0005)	1M (HV=0.74)	1M (HV=0.71)	1M (HV=0.66)
	PL-RA-E	672k ± 9k	**656k ± 18k**	*693k ± 16k* (p=0.1016)	*754k ± 24k* (p=0.1163)
Inequality rule	IQ-RA1	843k ± 15k (p=0.0015)	828k ± 20k (p=0.0115)	**822k ± 23k** (p=0.0169)	851k ± 8k (p=0.0325)
	IQ-RA2	836k ± 12k (p=0.0039)	838k ± 19k (p=0.0248)	**803k ± 18k** (p=0.0465)	834k ± 6k (p=0.0318)
	IQ-RA3	839k ± 21k (p=0.0036)	**819k ± 29k** (p=0.0219)	826k ± 10k (p=0.0454)	837k ± 5k (p=0.0414)
	IQ-RA-E	840k ± 27k (p=0.0024)	816k ± 22k (p=0.0351)	**798k ± 13k** (p=0.0311)	843k ± 11k (p=0.0223)
Mixed	PL-RA2 + IQ-RA2	*682k ± 10k* (p=0.0669)	***677k ± 14k*** (p=0.0816)	*691k ± 8k* (p=0.0772)	*751k ± 10k* (p=0.0522)
	PL-RA-E + IQ-RA-E	*676k ± 9k* (p=0.0714)	***659k ± 12k*** (p=0.0814)	*696k ± 13k* (p=0.0611)	*752k ± 23k* (p=0.0699)



(a) Pareto Front for one run.



(b) HV plot over 20 runs.

Figure 5.10 Pareto fronts and hypervolume plots obtained by IK-EMO with RU2 and power law repair agents for 1100-member truss design problem.

The observations from the results are outlined below.

5.8.2.1 General observations

Base NSGA-II is outperformed by all repair operators except for PL-RA3, which fails to achieve the target HV in most cases. This shows PL-RA3 being a subpar repair operator introducing too much uncertainty into the optimization process compared to the others. However, despite the low population size (almost 1/10th the number of variables), the repair operators are able to generate better solutions on an LSMOP. FE savings upto 25% compared to base NSGA-II is obtained.

5.8.2.2 Power law vs inequality rules

Similar to the previous two problems, power law repair operators are the best performer for each RU. Inequality rule-based repair operators in general result in a worse performance compared to power law-based repair operators except for the PL-RA3 case. These results show that power laws are in general a better choice compared to inequality rules.

5.8.2.3 Best performing algorithm for each RU

PL-RA-E is the best performer for RU1 and RU2, with PL-RA2 showing a comparable performance. For RU3 and RU4, PL-RA2 is the best performer, with PL-RA-E showing a comparable performance. PL-RA3 fails to achieve the target. For RU4, PL-RA1 is the best, with PL-RA2 and PL-RA-E giving statistically similar performance. As in the case of the previous problem, PL-RA-

E is either the best or gives a statistically similar performance to the best. PL-RA2 performance shows that an intermediate amount of trust in the learned power law is the optimal choice.

Figure 5.10 illustrates the results with RU2 combined with the power law repair operators for both problem cases. The difference in the quality of solutions obtained after 1 million FEs is more prominent than the previous two problems due to this being an LSMOP. PL-RA-E gives the best solutions are PL-RA3 is significantly worse. In the median HV plots it is seen that the FEs required to reach HV^T for PL-RA-E is close to PL-RA2. Base NSGA-II, PL-RA2 and PL-RA3 show a worse performance than the others.

5.8.2.4 Relative performance of each RU

It can be seen from Table 5.12 that RU2 gives the best performance in 5 out of 10 cases and RU3 gives the best performance in 4 out of 10 cases. This reinforces the previous results by showing that a moderate amount of rule usage gives better performance.

5.8.2.5 Mixed relation repair agents

All of the mixed relation repair agents (PL-RA2+IQ-RA2) and (PL-RA-E+IQ-RA-E), have a comparable performance to the best algorithm for each RU. Similar to previous problems, the worse performance of PL-RA3 and inequality repair operators do not affect the performance.

5.9 Summary of results

For every problem we have a total of 11 different algorithms including base NSGA-II and 4 RUs. Table 5.13 presents a ranking of all the algorithms for each RU based on the FEs required to reach the target HV. An algorithm with statistically similar performances to the best is given a rank of 1. The results are collected and an overall ranking is assigned to each algorithm. It is seen that the top ranked algorithm is PL-RA-E followed by PL-RA2+IQ-RA2, PL-RA-E+IQ-RA-E and PL-RA2. If the ideal repair agent or rule types are not known beforehand, the ensemble operator with mixed relationships (PL-RA-E+IQ-RA-E) can be used without significant performance drop.

5.10 Synchronous vs asynchronous user interaction

In the previous sections, it is assumed that user's feedback will be available soon after the learned rules are presented to the user. However, in the real world, users will always take some finite

Table 5.13 Ranking of different repair agents on multiple problems.

RU	Base	PL-RA1	PL-RA2	PL-RA3	PL-RA-E	IQ-RA1	IQ-RA2	IQ-RA3	IQ-RA-E	PL-RA2 +IQ-RA2	PL-RA-E +IQ-RA-E
78-variable stepped beam design											
RU1	4	3	1	3	1	5	2	5	4	1	1
RU2	4	2	1	2	1	3	2	4	3	1	1
RU3	5	4	1	2	1	5	3	4	5	1	1
RU4	2	2	1	5	1	4	3	2	2	1	1
Rank Sum	15	11	4	12	4	17	10	15	14	4	4
Rank	9	6	**1**	7	**1**	11	5	9	8	**1**	**1**
118-variable stepped beam design											
RU1	7	2	1	3	1	5	6	4	5	1	1
RU2	8	1	2	3	1	5	4	7	6	1	1
RU3	7	2	1	3	1	3	4	6	5	1	1
RU4	9	3	2	4	1	8	5	7	6	1	1
Rank Sum	31	8	6	13	4	21	19	24	22	4	4
Rank	11	5	4	6	**1**	8	7	10	9	**1**	**1**
115-variable optimal power flow											
RU1	3	2	1	5	1	4	4	3	3	1	1
RU2	5	2	1	3	1	7	6	4	3	1	1
RU3	7	3	1	5	1	4	6	4	8	1	2
RU4	2	1	1	4	1	4	3	5	4	1	1
Rank Sum	17	8	4	17	4	19	19	16	18	4	5
Rank	7	5	**1**	7	**1**	10	10	6	9	**1**	4
243-variable optimal power flow											
RU1	7	1	2	8	1	3	6	5	4	1	1
RU2	7	2	1	7	1	3	5	4	6	1	1
RU3	7	4	3	10	1	9	8	6	2	5	1
RU4	3	2	1	7	1	5	4	8	6	1	1
Rank Sum	24	9	7	32	4	20	23	23	18	8	4
Rank	10	5	3	11	**1**	7	8	8	6	4	**1**
1179-variable truss design											
RU1	7	2	1	8	1	6	3	4	5	1	1
RU2	7	2	1	8	1	5	6	4	3	1	1
RU3	7	2	1	8	1	5	4	6	3	1	1
RU4	7	2	1	8	1	6	3	4	5	1	1
Rank Sum	28	8	4	32	4	22	16	18	16	4	4
Rank	10	5	**1**	11	**1**	9	6	8	6	**1**	**1**
Final Rank Sum	47	26	10	42	5	45	36	41	38	8	8
Final Rank	11	5	4	9	**1**	10	6	8	7	2	2

time to come up with a preferred ranking of the rules. Pausing the optimization (synchronous user interaction) until the user provides feedback may be inefficient for problems with expensive function evaluations. Continuing the optimization tasks while the user finalizes their feedback (asynchronous user interaction) is a practicality, and seems like a more intriguing approach. We implement a simplistic asynchronous scenario here to investigate the effect of delayed feedback from users.

We consider both 118- and 300-bus OPF problems for this purpose. The user feedback lag (T_U) is expressed as the number of FEs that could have been executed between the time the user is presented with a set of rules and the time when the user is ready with some feedback (ranking of rules). For simplicity, T_U is assumed to be constant for every round of user interaction. In the synchronous user interaction case, T_U is undefined, as the optimization is put on hold until the user comes up with a ranking of the rules. The learning interval (T_L) – number of FEs executed between two consecutive rule learning tasks – and repair interval (T_R) – number of FEs executed between two consecutive repair operations – are set to 500 FEs, (with 50 population members, this means after every 10 generations). When the user provides feedback—ranking of the previous rule set provided to them—new rules having identical structure to previous rules are given priority, and the rest are discarded. But, instead of using previous rules' statistics (means and standard deviations of c, for example), statistics of the new rules are used to repair. The optimization with repair operations then proceeds with the updated preferred ranking of rules.

In the asynchronous user interaction case, once a VRG is constructed for the first time, IK-EMO is ready to provide information to the user about the learned rules every T_L FEs if the user is available. However, we let the optimization run proceed normally without waiting for user's feedback and perform a repair operation using the learned rules. After T_U FEs, the user is ready to provide feedback. Then, the user feedback on the immediate past rules is combined with the latest learned rules, as in the synchronous case, and repair is performed using the common rules but with latest rules' statistics. If $T_U \leq T_L$, the learned rules available to IK-EMO will be the same as the ones provided to the user before. Repair is performed immediately after the user provides the feedback. If $T_U > T_L$, the latest learned rules may be different than the ones the user was provided. It is to be noted that in the asynchronous case, T_R is not an adjustable parameter, since repair is performed as soon as one learning phase is complete, or the user has provided some feedback. A figure illustrating both processes in detail is provided in the supplementary document. In this study, we use different T_U values from 125 to 500 FEs ($T_U \leq T_L$) and from 1,000 to 4,000 FEs ($T_U > T_L$).

Table 5.14 FEs required to achieve HV^T for IEEE 118 and 300-bus OPF problems over 20 runs with multiple instances of asynchronous user interaction. Best performing algorithm for each row is marked in bold. Statistically similar results to the best are marked in italics.

Problem	HV^T	Base NSGA-II	Decision-making lag (T_U)					
			125	250	500	1000	2000	4000
IEEE 118-bus	0.74	6.5k ± 0.6k	**4.2k ± 0.1k**	*4.4k ± 0.3k*	*4.5k ± 0.4k*	*4.8k ± 0.4k*	5.2k ± 0.2k	6.0k ± 0.5k
IEEE 300-bus	0.70	17.5k ± 1.2k	**14.4k ± 0.3k**	*14.5k ± 0.1k*	*14.4k ± 0.5k*	*14.7k ± 0.4k*	15.1k ± 0.2k	16.0k ± 0.3k

Table 5.15 Final median HV obtained for IEEE 118 and 300-bus OPF problems over 20 runs with multiple instances of synchronous user interaction. Best performing algorithm for row is marked in bold. Algorithms with statistically similar performance to the best performing algorithm are marked in italics. Results are presented graphically in the supplementary document.

Problem	T_c	T_U = 125		T_U = 250		T_U = 500		T_U = 1,000		T_U = 2,000		T_U = 4,000	
		Sync	Async	Sync	Async	Sync	Async	Sync	Async	Sync	Async	Sync	Async
IEEE 118-bus	20k	**0.86**	*0.84*	**0.86**	*0.84*	0.80	0.79	0.74	0.79	0.69	0.77	0.51	0.76
IEEE 300-bus	20k	**0.84**	**0.84**	*0.83*	*0.81*	0.74	*0.82*	0.71	*0.80*	0.66	*0.80*	0.43	0.77

Table 5.14 shows that for both IEEE 118-bus and 300-bus systems, a lag of up to 1,000 FEs gives statistically similar performance to the synchronous case. For lags of 2,000 FEs or higher, the performance deteriorates. But even with large lags, IK-EMO is robust enough to perform better than base NSGA-II. A quicker user feedback with $T_U \leq T_L$ produces the optimal performance, as expected. With a large lag time for user decision, feedback based on old rules has a detrimental effect. Thus, asynchronous interaction is suitable for cases where the objective evaluation is overly expensive, providing users relatively more time to make a decision on preferential ranking of rules.

A second study evaluates the effect of user feedback lag time for a fixed overall computational time of T_c units. For the synchronous case, the effective number of FEs allocated for the optimization operations becomes small, since a part of T_c is now consumed by the user to make a decision. For multiple values of T_U, we compare the final HV obtained for asynchronous interaction with non-zero lag cases with that of the synchronous user interaction case. Table 5.15 shows such a comparison with lag values varying from 125 to 4,000 FEs for a fixed overall execution time of $T_c = 20,000$ FEs.

From the results, we observe that the performance of the synchronous case drops drastically when T_U is increased for both 118- and 300-bus OPF problems. This is because for a large lag in making decisions, more time is wasted in the decision-making and less execution time is provided for running the optimization operations. For the asynchronous case, the performance drops slowly with T_U, since IK-EMO is able to use the full computational budget of 20k FEs for both 118- and 300-bus cases. This outweighs any performance loss caused by using outdated rules.

From the results presented in this section, it is evident that user feedback lag is an important practical factor that will have an effect in an interactive optimization procedure. These preliminary results suggest that for a small lag, both synchronous and asynchronous implementations can be viable options. However, for a large anticipated lag, asynchronous implementation is better.

CHAPTER 6

IK-EMOVIZ: A SOFTWARE IMPLEMENTATION FOR THE IK-EMO FRAMEWORK

IK-EMO Visualizer, or IK-EMOViz is a software implementation of user interactivity in the IK-EMO framework. In this section, we briefly cover the IK-EMOViz implementation shown in Figure 6.1.

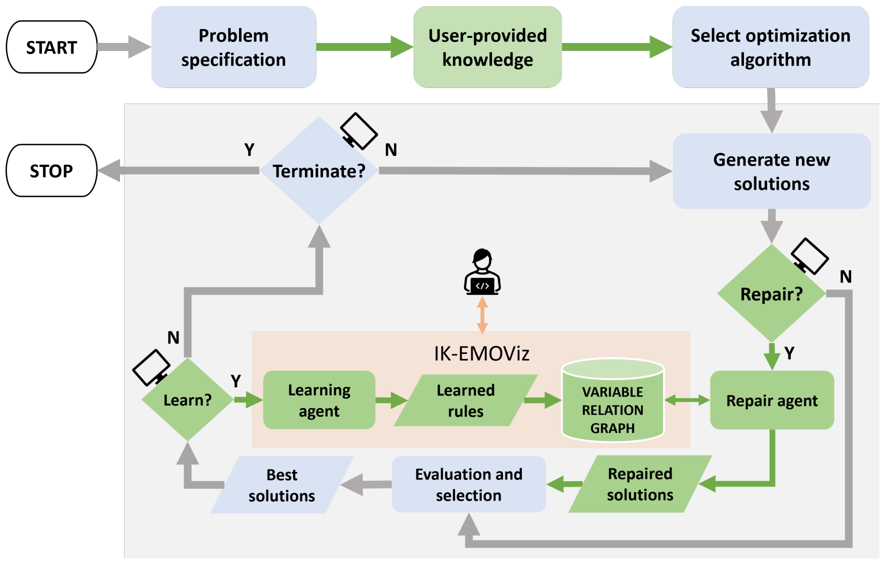


Figure 6.1 Interactive knowledge-based EMO framework (IK-EMO). Blue blocks represent a normal EMO. Green blocks represent the automated knowledge extraction and application. Information from the learning agents and variable relation graphs is presented to the user using IK-EMOViz. The user, in turn, can provide feedback using the same interface.

6.1 User interaction using the IK-EMOViz graphical user interface

IK-EMOViz is implemented in Python using Plotly and Dash [120], which provides a browser-based graphical user interface (GUI) for the IK-EMO framework. IK-EMOViz allows the user to access real-time data about the optimization run such as convergence indicators, scatter plots, and parallel coordinate plots (PCP) through separate widgets. Figure 6.2 shows an example instance of IK-EMOViz where the user wants to analyze the results of a bi-objective optimization problem after 60 generations. The optimization progress in this case is represented by a hypervolume (HV) [104] evolution plot (left-most plot in Figure 6.2) over the generations completed. The plot is dynamically updated as each generation is completed. Indicators other than HV can also be used, if desired. The scatter plot widget (middle plot) shows the entire population in the objective space

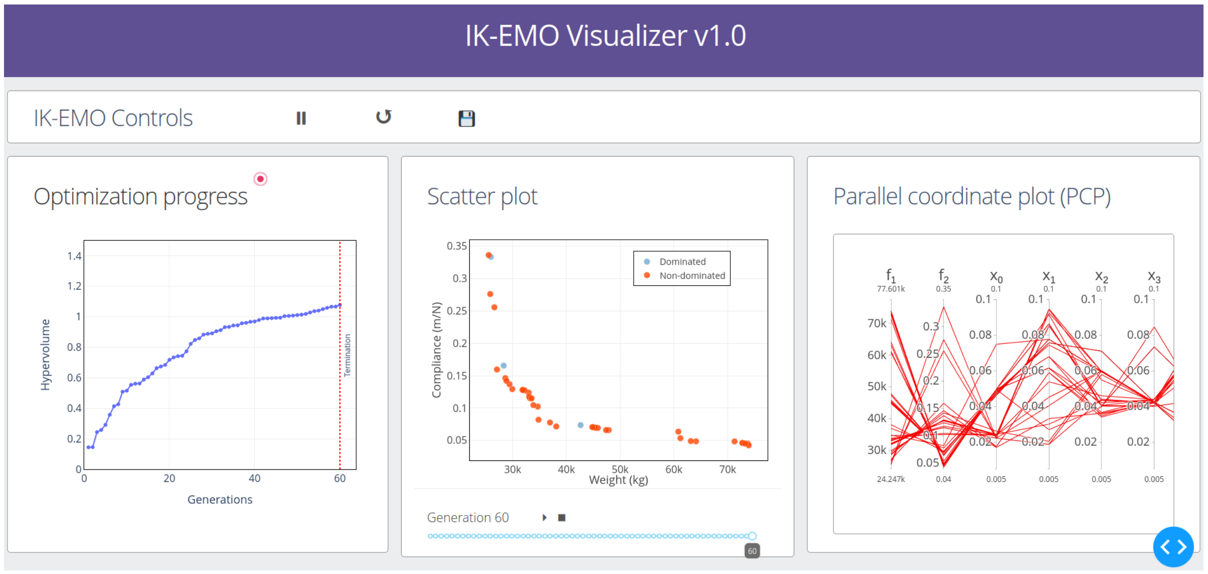
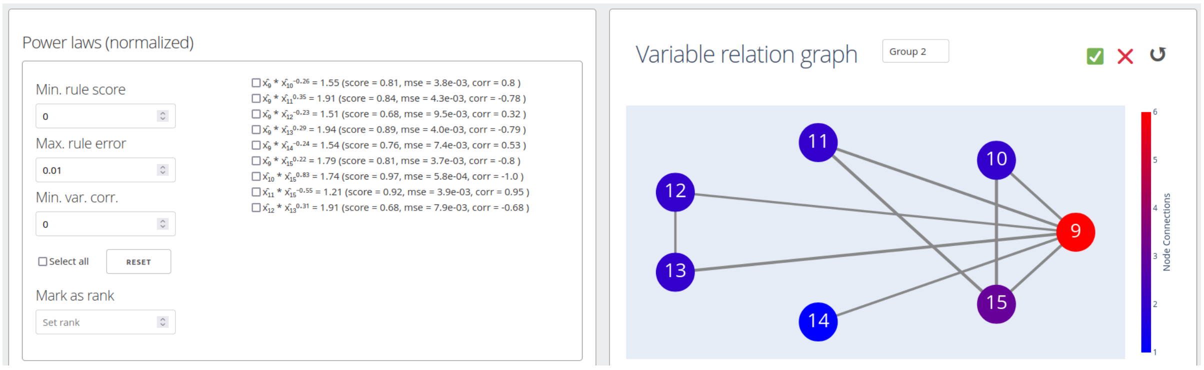


Figure 6.2 IK-EMOViz graphical user interface showing the optimization progress, scatter plots, and parallel coordinate plot widgets describing the results of a bi-objective optimization problem which was terminated after 60 generations.

with the non-dominated solutions marked in orange and dominated solutions marked in blue. By using the generator slider shown below the plot, the user can also check the objective vectors of the population in any earlier generation. The software saves all earlier populations and can display any earlier population, if desired. The PCP plot (right-most plot) gives a visualization of the objective and decision variable values together. An additional widget is available whereby the user can visualize any solution from the Pareto front scatter plot, but for brevity, it is not shown here.

The ‘IK-EMO Controls’ widget has three buttons (not shown here). The first button can pause or resume the optimization run. This is used if the user wants the algorithm to wait till he/she analyzes the results and provides feedback, also known as ‘synchronous interaction’. The second button is used to refresh all the widgets except the one showing optimization progress to get the latest data. This functionality is necessary if the user did not pause the optimization run (‘asynchronous interaction’) and wants to see the updated results and their associated rules. The third button saves any user feedback which will be considered by IK-EMO in subsequent generations.

Another important widget displays the latest rules and the corresponding VRGs generated by the learning agent for all variable groups. A group selector menu allows the user to switch among multiple groups. Figure 6.3 shows the rule list and VRG for Group 2 variables of the example problem. Two variables found to possess a significant relation are connected by a gray edge whose thickness is proportional to the rule score. Nodes are colored on the basis of their degree, with reddish nodes indicating a high number of connected edges, and bluish nodes representing a low number of connected edges.


Power laws (normalized)

Min. rule score: 0

Max. rule error: 0.01

Min. var. corr.: 0

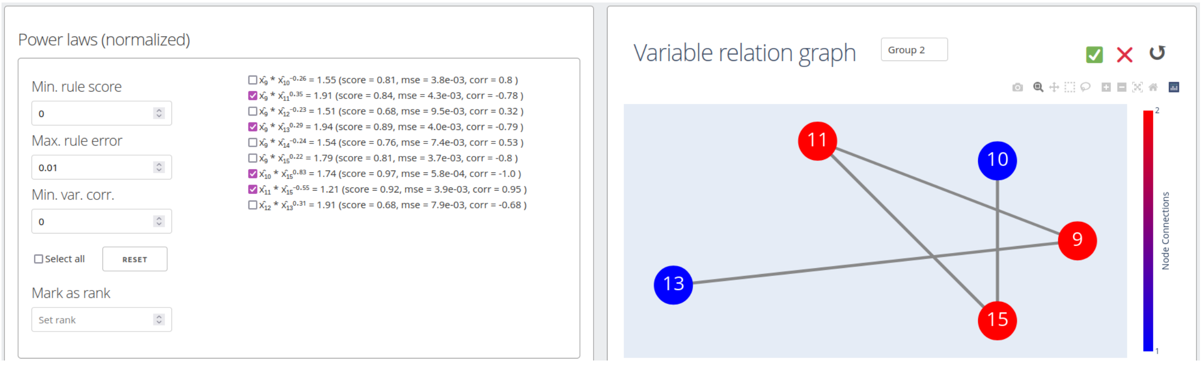
- ☐ $\hat{x_9} * \hat{x_{10}}^{-0.26} = 1.55$ (score = 0.81, mse = 3.8e-03, corr = 0.8)
- ☐ $\hat{x_9} * \hat{x_{11}}^{0.35} = 1.91$ (score = 0.84, mse = 4.3e-03, corr = -0.78)
- ☐ $\hat{x_9} * \hat{x_{12}}^{-0.23} = 1.51$ (score = 0.68, mse = 9.5e-03, corr = 0.32)
- ☐ $\hat{x_9} * \hat{x_{13}}^{0.29} = 1.94$ (score = 0.89, mse = 4.0e-03, corr = -0.79)
- ☐ $\hat{x_9} * \hat{x_{14}}^{-0.24} = 1.54$ (score = 0.76, mse = 7.4e-03, corr = 0.53)
- ☐ $\hat{x_9} * \hat{x_{15}}^{0.22} = 1.79$ (score = 0.81, mse = 3.7e-03, corr = -0.8)
- ☐ $\hat{x_{10}} * \hat{x_{15}}^{0.83} = 1.74$ (score = 0.97, mse = 5.8e-04, corr = -1.0)
- ☐ $\hat{x_{11}} * \hat{x_{15}}^{-0.55} = 1.21$ (score = 0.92, mse = 3.9e-03, corr = 0.95)
- ☐ $\hat{x_{12}} * \hat{x_{13}}^{0.31} = 1.91$ (score = 0.68, mse = 7.9e-03, corr = -0.68)

Figure 6.3 IK-EMOViz graphical user interface showing the full VRG for variable group 2 for a specific optimization problem.

After analyzing the VRG and associated rules, the user provides feedback by modifying one or more nodes in the VRG. The following operations can be performed on the learned rules.

- Exclusion: The user may select to remove certain rules provided by the algorithm, based on their knowledge of the problem. The VRG will be updated by removing the corresponding edges.
- Selection: The user may wants to keep only certain rules. In that case, the corresponding VRG edges will be retained and the rest will be deleted.
- Filtering: If there are a large number of rules, the user can choose to filter them based on criteria like rule scores, variable correlations, etc.
- Ranking: The user may provide a ranking of rules (rank 1 is most preferred) provided by the algorithm. The algorithm will then try to implement the rules according to the ranks, as demonstrated in [121].

Figure 6.3 shows a portion of the IK-EMOViz GUI for the example problem considered previously. For Group 2, the list of power laws obtained is shown on the left and the corresponding VRG is shown on the right. Figure 6.4 shows the selected rules by the user achieved by clicking the corresponding check-boxes. On clicking the green tick button (top corner in right plot), the VRG is updated with only the selected rules. For example, Node 12 in Figure 6.3 is now absent in Figure 6.4, since the user did not select any rule involving variable x_{12}. This type of operation can be useful when the user only wants to select a few rules involving a few important variables.


Power laws (normalized)

Min. rule score: 0

Max. rule error: 0.01

Min. var. corr.: 0

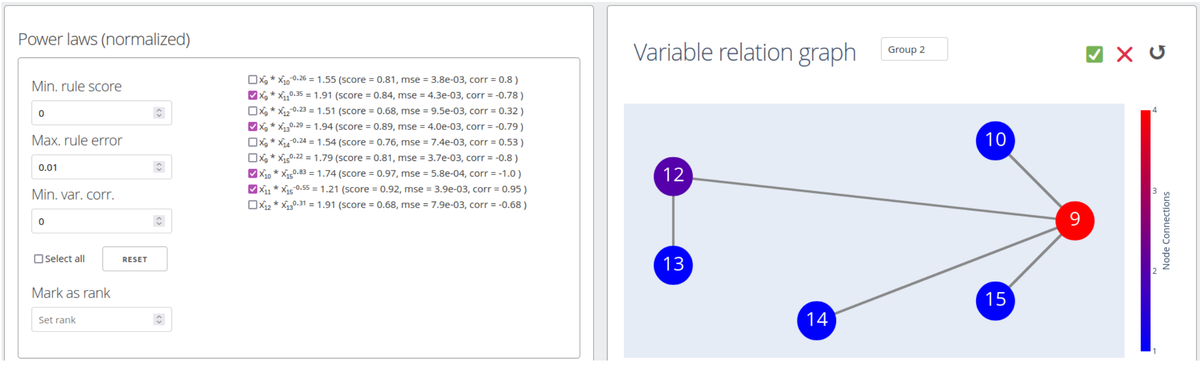
☐ Select all · RESET

Mark as rank: Set rank

☐ $\hat{x_9} * \hat{x_{10}}^{-0.26} = 1.55$ (score = 0.81, mse = 3.8e-03, corr = 0.8)
☑ $\hat{x_9} * \hat{x_{11}}^{0.35} = 1.91$ (score = 0.84, mse = 4.3e-03, corr = -0.78)
☐ $\hat{x_9} * \hat{x_{12}}^{-0.23} = 1.51$ (score = 0.68, mse = 9.5e-03, corr = 0.32)
☑ $\hat{x_9} * \hat{x_{13}}^{0.29} = 1.94$ (score = 0.89, mse = 4.0e-03, corr = -0.79)
☐ $\hat{x_9} * \hat{x_{14}}^{-0.24} = 1.54$ (score = 0.76, mse = 7.4e-03, corr = 0.53)
☐ $\hat{x_9} * \hat{x_{15}}^{0.22} = 1.79$ (score = 0.81, mse = 3.7e-03, corr = -0.8)
☑ $\hat{x_{10}} * \hat{x_{15}}^{0.83} = 1.74$ (score = 0.97, mse = 5.8e-04, corr = -1.0)
☑ $\hat{x_{11}} * \hat{x_{15}}^{-0.55} = 1.21$ (score = 0.92, mse = 3.9e-03, corr = 0.95)
☐ $\hat{x_{12}} * \hat{x_{13}}^{0.31} = 1.91$ (score = 0.68, mse = 7.9e-03, corr = -0.68)

Figure 6.4 IK-EMOViz graphical user interface showing a rule selection operation.

Figure 6.5 shows the resulting VRG when some rules are excluded from Figure 6.3 by selecting them in the rule list and clicking the cross button on the top right of the VRG. This is useful when the user only wants to exclude specific rules provided by the algorithm.


Power laws (normalized)

Min. rule score: 0

Max. rule error: 0.01

Min. var. corr.: 0

☐ Select all · RESET

Mark as rank: Set rank

☐ $\hat{x_9} * \hat{x_{10}}^{-0.26} = 1.55$ (score = 0.81, mse = 3.8e-03, corr = 0.8)
☑ $\hat{x_9} * \hat{x_{11}}^{0.35} = 1.91$ (score = 0.84, mse = 4.3e-03, corr = -0.78)
☐ $\hat{x_9} * \hat{x_{12}}^{-0.23} = 1.51$ (score = 0.68, mse = 9.5e-03, corr = 0.32)
☑ $\hat{x_9} * \hat{x_{13}}^{0.29} = 1.94$ (score = 0.89, mse = 4.0e-03, corr = -0.79)
☐ $\hat{x_9} * \hat{x_{14}}^{-0.24} = 1.54$ (score = 0.76, mse = 7.4e-03, corr = 0.53)
☐ $\hat{x_9} * \hat{x_{15}}^{0.22} = 1.79$ (score = 0.81, mse = 3.7e-03, corr = -0.8)
☑ $\hat{x_{10}} * \hat{x_{15}}^{0.83} = 1.74$ (score = 0.97, mse = 5.8e-04, corr = -1.0)
☑ $\hat{x_{11}} * \hat{x_{15}}^{-0.55} = 1.21$ (score = 0.92, mse = 3.9e-03, corr = 0.95)
☐ $\hat{x_{12}} * \hat{x_{13}}^{0.31} = 1.91$ (score = 0.68, mse = 7.9e-03, corr = -0.68)

Figure 6.5 IK-EMOViz graphical user interface showing a rule exclusion operation.

6.1.1 Synchronous vs asynchronous user interaction

Pausing the optimization until the user finishes providing feedback ensures the user always has access to the latest data. This is also referred to as synchronous interaction. However, the user can take a longer time to analyze the results and provide his/her feedback. It can be more practical to continue running the optimization in the background while the user analyzes the intermediate solutions and their associated rules. This is known as asynchronous interaction. In the previous section, the pause optimization functionality of IK-EMOViz was introduced. It allows the user to switch between synchronous and asynchronous interaction modes.

Apart from the learning interval (T_L) and repair interval (T_R), another quantity, the user feedback lag (T_U), was introduced. This is defined as the number of generations or SEs that can be performed during the time the user is analyzing the results and preparing the feedback. T_U is undefined for synchronous user interaction. Figure 6.6 illustrates the asynchronous user interaction mechanism of IK-EMOViz, assuming $T_L = T_R$. After every T_L SEs, the learning agent generates a new VRG for each variable group, marked by the blue vertical dashed lines. IK-EMOViz allows the user to provide feedback at any point thereafter during the optimization run. The green dashed lines represent a user interaction phase lasting for T_U SEs. In the example shown in Figure 6.6, the user launches IK-EMOViz between learning rounds 2 and 3. Since user interaction is active, no repair is performed during learning rounds 3 and 4. Once user feedback is complete, the user-modified VRG (VRG$^{(U)}$) is merged with the last learned VRG (VRG$^{(4)}$) as specified in Section 5.5.5, and the combined VRG is used by the repair agent. Thus, a larger T_U means the user is making a decision based on potentially outdated information and may not be as efficient as using the latest knowledge.

6.2 Truss Design Problem

For this study, we use the scalable truss optimization problem described in Chapter 4.

6.2.1 Experimental settings

In this study, we use two instances of the truss design problem, a truss with 120 members, 36 nodes, 129 decision variables and 156 non-linear constraints, and another larger truss with 820 members, 236 nodes, 879 decision variables, and 1056 non-linear constraints. Variable groups were created

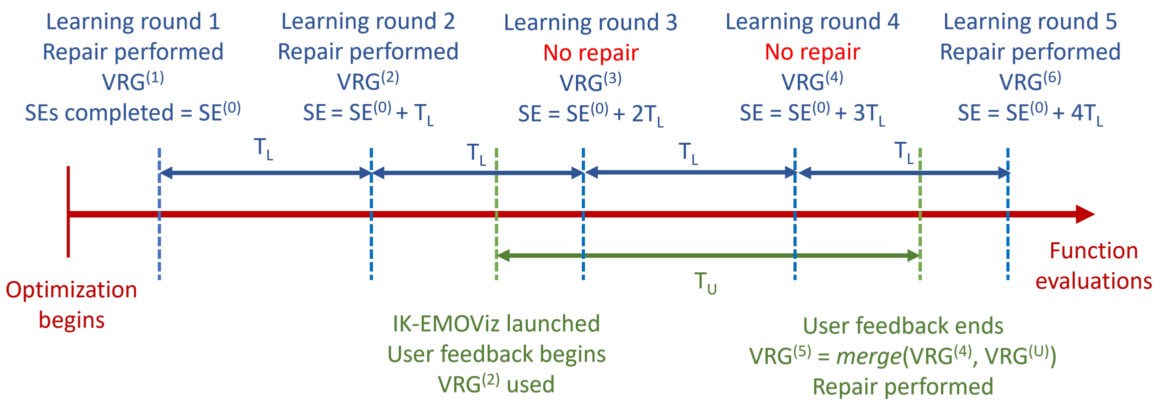


Figure 6.6 Asynchronous user interaction example with $T_L = T_R$. Blue dashed lines represent the learning phases taking place at intervals of T_L SEs. Green dashed lines represent a user interaction phase.

similar to [121] according to the physical orientation of the truss beams as shown in Table 6.1. The

Table 6.1 Variable groups for the 120 and 820-member truss design problems.

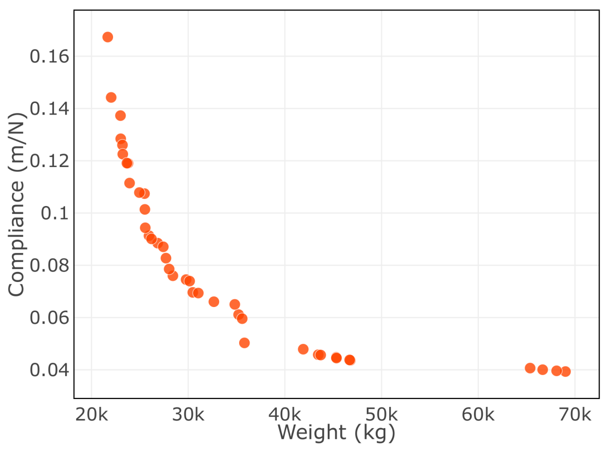
Group	Variable Type	Variable Indices	
		120-member truss	820-member truss
G_1	l_i of vertical members	[120 – 128]	[820 – 878]
G_2	r_i of top longitudinal members	[8 – 15], [24 – 31]	[0 – 57], [116 – 173]
G_3	r_i of bottom longitudinal members	[0 – 7], [16 – 23]	[58 – 115], [174 – 231]
G_4	r_i of vertical members	[36 – 53]	[236 – 353]

120-member truss is used to illustrate how IK-EMOViz can allow the user to obtain insights about an optimization problem. NSGA-II [15] is chosen as the optimization algorithm of IK-EMO in this study. Population size is set as 40 and the maximum number of generations is set as 100. After 100 generations, IK-EMOViz is launched. The results are presented in the next section.

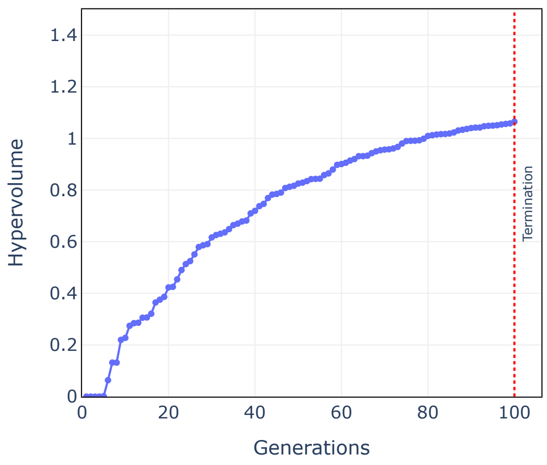
The 820-member truss design problem is used to illustrate the power of IK-EMO in finding good solutions with periodic user interactions. NSGA-II is run for a maximum number of generations of 12,500 (set by trial-and-error process and required to work with 879 variables), thus giving a total computational budget of 500,000. To ensure consistency, three artificial users – U1, U2, and U3 are created to select rules from the all the generated rules by our rule learning method with scores above 0.9, 0.7, and 0.5, respectively. Thus, user U1 chooses fewer rules compared to U2 and U3. For each user, four rule adherence schemes (RA1, RA2, RA3, and RA-E) are considered, making a total of 12 separate optimization runs. Moreover, two different scenarios are considered: synchronous interaction, where optimization is paused when user interaction takes place after $T_U = 10,000$ SEs would have elapsed, and asynchronous interaction, in which the optimization does not wait for during the analysis process by the user. Since we would like to complete the runs after a fixed execution time is achieved, the synchornous interaction cases are allocated less overall SEs (for each analysis mode, T_U SEs are doscounted). For each user-repair agent combination, 20 runs are performed, involving a total of 480 optimization runs. The hypervolume (HV) [104] metric value for each run is recorded.

6.2.2 Experimental results and discussion on the 120-member truss

For the 120-member truss, the Pareto front and HV evolution plots are shown in Figure 6.7. The figures are extracted from IK-EMOViz. From the Pareto-optimal solutions found till now, power



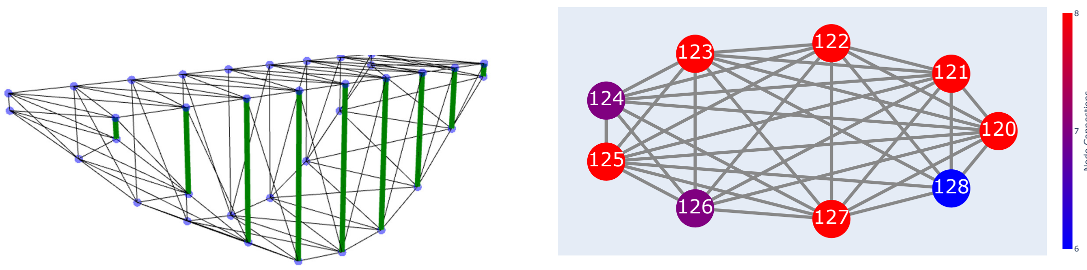
(a) Pareto front.



(b) Hypervolume vs. generations.

Figure 6.7 Pareto front and hypervolume plot for a 120-member truss optimization problem obtained from IK-EMOViz.

law rules are extracted. Group G_1 representing the length of the vertical members are shown in Figure 6.8 highlighted in green. From Figure 6.8b, it can be seen that all the variables are related to each other through some significant power law. However, through the functionality of IK-EMOViz we can perform some simplifications. Variables 120-128 represent consecutive vertical members shown in Figure 6.8a from right to left. So we perform rule selection and select rules of the form



(a) 120-member truss. (b) Variable relation graph.

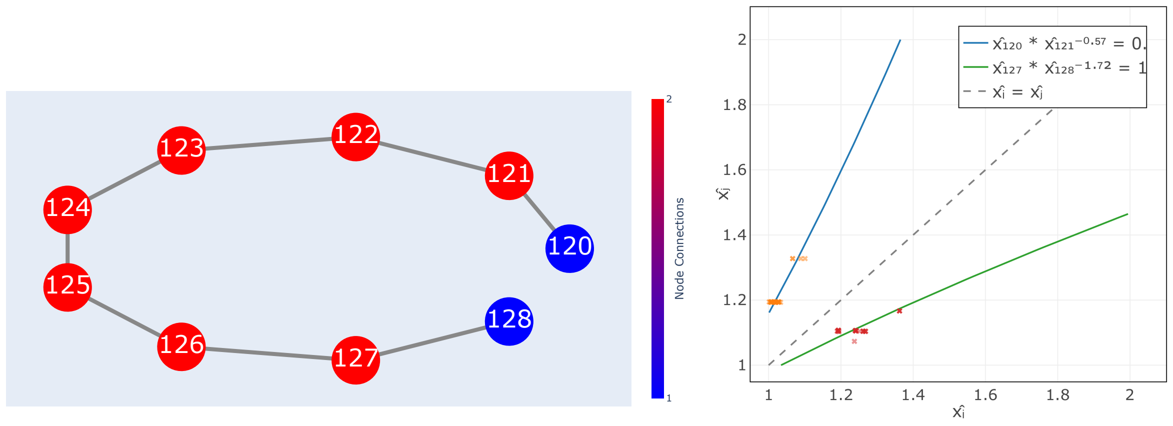
Figure 6.8 A Pareto-optimal truss design with group G_1 highlighted in green and the corresponding variable relation graph obtained from IK-EMOViz.

$x_i x_{i+1}^b = c$ where $i = 120, 121, \ldots, 127$. The reduced set of power law rules thus obtained for G_1 along with their rule compliance values are given in Table 6.2. Rule compliance is defined as the proportion of non-dominated solutions that follow a power law. The corresponding VRG is shown in Figure 6.9a.

Table 6.2 Power law rules found for group G_1.

Rule no.	Power law	Rule compliance
1	$x_{120}x_{121}^{-0.57} = 0.92$	1.00
2	$x_{121}x_{122}^{-1.00} = 0.87$	1.00
3	$x_{122}x_{123}^{-0.92} = 0.96$	1.00
4	$x_{123}x_{124}^{-0.82} = 1.05$	1.00
5	$x_{124}x_{125}^{-1.13} = 0.98$	1.00
6	$x_{125}x_{126}^{-0.78} = 1.14$	1.00
7	$x_{126}x_{127}^{-1.06} = 1.10$	1.00
8	$x_{127}x_{128}^{-1.72} = 1.03$	1.00

From Table 6.2, let us consider two power laws $\phi_1(\mathbf{x}) = x_{120}x_{121}^{-0.57} - 0.92 = 0$ and $\phi_2(\mathbf{x}) = x_{127}x_{128}^{-1.72} - 1.03 = 0$. x_{120} and x_{128} represent the length of the vertical members at each end, and x_{121} and x_{127} represent the adjacent vertical members, respectively. Figure 6.9b obtained from IK-EMOViz plots the two power laws over the normalized variable range of $[1, 2]$ and plots another line $x_i = x_j$. The non-dominated solutions from which the power laws where extracted are also shown. An interesting observation is that the $\phi_1(\mathbf{x})$ lies above and $\phi_2(\mathbf{x})$ lies below the $x_i = x_j$ line. This indicates that $x_{120} < x_{121}$ and $x_{127} > x_{128}$ among good solutions. Analyzing the rest of the power laws in Table 6.2 and the loading condition of the truss, we can see that $x_{120} < x_{121} < \ldots < x_{124}$



(a) Simplified variable relation graph.

(b) Power laws.

Figure 6.9 Simplified variable relation graph of group G_2 and power laws between x_{120}, x_{121} and x_{127}, x_{128} obtained from IK-EMOViz.

and $x_{124} > x_{125} > \ldots > x_{128}$. This indicates that the length of the vertical members increase from the end to the middle of the truss, which is an expected outcome of this particular loading condition [92]. IK-EMO is able to successfully extract these rules and IK-EMOViz provides useful functionality to the user to understand the rules.

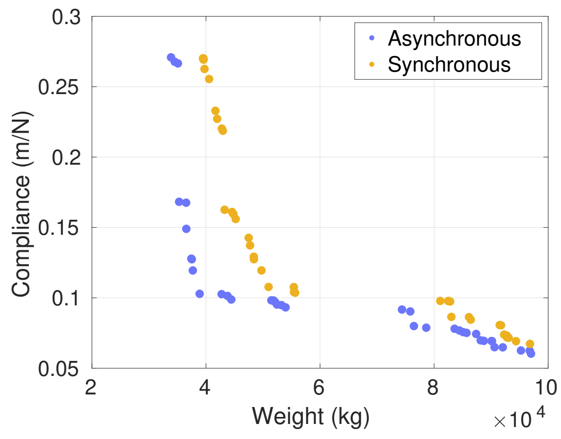
6.2.3 Experimental results and discussion on the 820-member truss

The final median HV obtained at the end of 500k SEs for each user-repair agent combination are shown in Table 6.3. The row-wise best is marked in bold with the statistically similar performing algorithms marked in italics. The column-wise best is marked by a gray box. It can be seen that user U2 who selects a moderate amount of rules is the best performer for all the repair agents. For each user, RA2 is the best performer in 4 out of 6 cases, and RA-E has a the best performance in 3 out of 6 cases. This shows that an intermediate amount of rule usage combined with a medium level of rule adherence works best, as shown in [121]. In all the cases, asynchronous interactions results in a better performance due to a high T_U. While asynchronous interaction runs the risk of the user making a decision based on outdated rules, the built-in safeguards of IK-EMO against incorrect user information [121] mitigate some of the risks. Thus, allowing the optimization to run in the background while the user deliberates on the available data is a better approach compared to pausing the optimization. Thus, a tool like IK-EMOViz which provides the option to the user to keep running the optimization in the background is necessary.

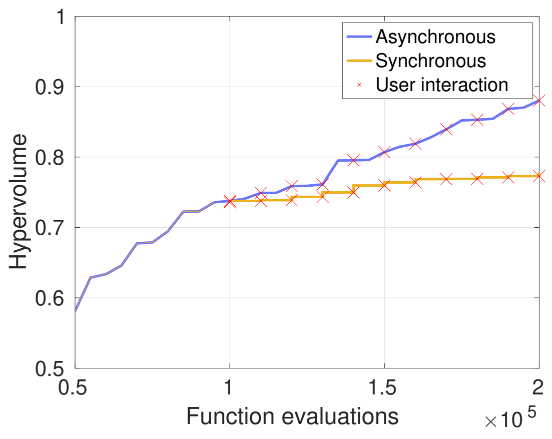
Table 6.3 Final median HV obtained at the end of 500k SEs for the 820-member truss optimization problem. Best-performing algorithm in each row is marked in bold. Statistically similar performance to the best in each row is marked in italics. The gray boxes represent the column-wise best.

Repair agent	U1		U2		U3	
	Sync	Async	Sync	Async	Sync	Async
None (base)	0.79	0.79	0.79	0.79	0.79	0.79
RA1	0.81	0.98	0.88	**1.01**	0.82	0.95
RA2	0.84	*1.06*	0.91	**1.08**	0.77	0.99
RA3	0.81	0.96	0.86	**0.98**	0.79	0.90
RA-E	0.85	1.02	0.88	**1.08**	0.79	0.99

For user U2 and repair agent RA2, the PF and HV plots comparing synchronous and asynchronous interaction cases are shown in Figure 6.10. Figure 6.10a shows how asynchronous interaction can provide better solutions compared to synchronous interaction. The HV plot in Figure 6.10b shows a portion of the optimization run between 50k and 200k SEs with user interaction instances being marked in red. Asynchronous interaction results in a better median HV.



(a) Pareto Front for one run.



(b) Median HV plot over 20 runs.

Figure 6.10 Pareto fronts and median hypervolume plots obtained by IK-EMO with U2, RA2, and asynchronous interaction for 820-member truss design problem.

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

7.1 Contributions of this dissertation

This dissertation describes the process of converting a basic EMO algorithm into an interactive method which uses automated knowledge-extraction methods. Knowledge, in this context, is defined as mathematical relations among two or more variables such as power laws, inequalities, constant rules, etc.

The first framework (MOEA/I) extracts inequality and equality rules during the optimization run. This knowledge is applied in the form of repair operators, where newly-generated solutions are modified according to any learned rules. Three such repair operators are introduced, along with an ensemble method capable of combining multiple repair operators and adaptively shifting between them based on their performance. The performance is demonstrated on a large-scale truss design problem (879 and 1479 variables), as well as a solid rocket design problem (544 variables). MOEA/I is shown to provide better solutions overall as well as to exhibit faster convergence on all the problems.

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

The IK-EMO framework has been applied to multiple large-scale two-objective practical problems: 78- and 118-variable stepped beam design problems, 115- and 243-variable optimal power flow problems, and a 1,179-variable truss design problem. Experimental results on all problems have consistently shown that (i) usage of a moderate number of rules (20%) combined with a moderate degree of rule adherence produces the best performance, and (ii) power law rules, individually, produce better performance than inequality rules. Moreover, ensemble-based repair operators provide comparable performance to the best performing individual repair operators. Use of ensembles eliminates the need to experiment to find the right rule adherence for a new problem. IK-EMO is also able to work with very low population sizes, even for a large-scale problem.

A preliminary study on a practical aspect—the inevitable lag time between presenting learned rules to the user and obtaining feedback from the user—has been made. Results on the OPF problem have shown that the proposed framework is able to maintain similar performance up to a certain lag period, beyond which the user response has been found to be too slow for the algorithm to maintain the same level of performance.

This study has introduced a knowledge-based interactive optimization tool IK-EMOViz for executing a better-informed optimization study. Through a 120-member truss design problem, we have shown how IK-EMOViz can help the user to obtain useful insights about an optimization problem in the form of simplistic relationships among variables and visualization of the interaction through relationship graphs. In addition, the user can also provide their own feedback by providing their preferences on obtained relationships using the tool. IK-EMOViz also allows the user to perform an asynchronous interaction by utilizing computing resources in the background while understanding and analyzing the obtained relationships. As demonstrated on an 820-member truss problem, the option of asynchronous interaction can result in a better performance. Moreover, moderately few relationships chosen by the user applied with moderate adherence within the EMO algorithm have found to produce better results.

7.2 Future work

This study opens up a number of avenues for future work, discussed below.

7.2.1 Varying learning and repair intervals

In this work, the learning and repair intervals are kept fixed. The effect of these parameters need to be studied more closely. Learning every 10 generations can result in different results compared to learning every generation. Frequent repair operations can result in a much more accurate probability update for ensemble operations. On the other hand, too many repairs done within a short period of time can steer the optimization algorithm towards undesired regions in the search space.

7.2.2 Locality of learned rules

In this study, the learning agents operated on the assumption that learned rules are of high quality if they are applicable across the entire Pareto front. Some rules may exist in only certain parts of the Pareto-optimal front. The existence of transition points [67] is one such example. Exploring ways for IK-EMO to extract local rules and repair offspring population members accordingly will introduce additional challenges but may result in faster convergence. Appropriate evaluation metrics also need to be developed.

7.2.3 Integration of traditional user preference incorporation methods

This study uses a novel way to integrate user-provided knowledge into an optimization framework. Traditional user preference information including, but not limited to, aspiration levels, relative importance of objective functions, and preferred regions of the Pareto-optimal front, can also potentially be integrated into IK-EMO.

7.2.4 Asynchronous user interaction

The asynchronous user interaction study is practical and must be investigated more thoroughly. One future direction would be to study the effect of variable user interaction lag (T_U). This is relevant since in a practical setting, the user is unlikely to take the same amount of time for every round of interaction. The user is also not likely to interact as frequently as assumed in this study. Thus, studies comparing synchronous and asynchronous user interaction modes need to be more oriented towards practical situations.

7.2.5 Improvements to IK-EMOViz

Future work can focus on enhancing the functionality of IK-EMOViz by allowing direct modification of rule parameters. The user should also be able to introduce new rules if they are not present in the rule set extracted by the learning agents. Other modifications can include introduction of new repair agents or disabling existing ones. The VRG widget can also be made interactive by turning it into a 3D model, allowing the user to directly modify the VRG. Nevertheless, this study provides evidence of the importance of a robust software implementation for any knowledge-based interactive optimization method that combines human knowledge and machine intelligence in executing an optimization task faster than either of them alone.

7.3 Concluding remarks

Designers often have years of knowledge and intuition in dealing with the problem they are interested to optimize. Innovization methods also exist which allows us to extract this knowledge in an online fashion. In this work, we attempt to provide an EMO-based framework which is able to leverage both human-provided as well as computer-extracted problem knowledge to achieve faster convergence. The proposed framework is able to outperform NSGA-II on large scale problems ranging up to 1,479 variables. The work in this dissertation demonstrates the potential of developing and applying interactive knowledge-driven optimization methods in practical applications.

BIBLIOGRAPHY

[1] M. O. Grond, N. H. Luong, J. Morren, and J. G. Slootweg, "Multi-objective optimization techniques and applications in electric power systems," 2012.

[2] J. L. J. Pereira, G. A. Oliver, M. B. Francisco, S. S. Cunha, and G. F. Gomes, "A review of multi-objective optimization: Methods and algorithms in mechanical engineering problems," *Archives of Computational Methods in Engineering*, vol. 29, pp. 2285–2308, 6 2022.

[3] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, pp. 115–148, 1995.

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

[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, pp. 460–474, 9 2017.

[6] 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. Semi independent vars paper<br/>Useful for innovization.

[7] K. Deb and A. Srinivasan, "Innovization: Innovating design principles through optimization," vol. 2, pp. 1629–1636, 2006.

[8] S. Bandaru and K. Deb, "Higher and lower-level knowledge discovery from pareto-optimal sets," vol. 57, pp. 281–298, Springer, 10 2013.

[9] A. Gaur and K. Deb, "Adaptive use of innovization principles for a faster convergence of evolutionary multi-objective optimization algorithms," pp. 75–76, Association for Computing Machinery, Inc, 7 2016.

[10] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, vol. 5. Springer US, 2007.

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

[12] K. Miettinen, "Nonlinear multiobjective optimization," vol. 12, 1998.

[13] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization," 1993.

[14] E. . Zitzler, L. Thiele, and E. Zitzler, "An evolutionary algorithm for multiobjective optimization: the strength pareto approach," 1998.

[15] 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, pp. 182–197, 4 2002.

[16] J. Nocedal and S. Wright, "Solutions to the selected problems in numerical optimization," p. 524, 2006.

[17] D. Goldberg, "Genetic algorithms in search optimization and machine learning," 1988.

[18] J. Koza, "Genetic programming: on the programming of computers by means of natural selection," *Biosystems*, vol. 33, pp. 69–73, 1992.

[19] H.-P. P. Schwefel, *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley &amp; Sons, Inc., 1993.

[20] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 712–731, 12 2007.

[21] E. . Zitzler, M. . Laumanns, L. Thiele, E. Zitzler, and M. Laumanns, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK Report*, vol. 103, 2001.

[22] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 577–601, 2014.

[23] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, 2014.

[24] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results.," *Evolutionary computation*, vol. 8, pp. 173–195, 3 2000. ZDT original.

[25] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," vol. 1, pp. 825–830, IEEE Computer Society, 2002. DTLZ original.

[26] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," vol. 3410, pp. 280–295, Springer Verlag, 2005. WFG original.

[27] Q. Zhang, A. Zhou, and S.-Z. Zhao, "Multiobjective optimization test instances for the cec 2009 special session and competition," *University of Essex*, pp. 1–30, 2008. CEC 2009 original.

[28] K. Deb and C. Myburgh, "Breaking the billion-variable barrier in real-world optimization using a customized evolutionary algorithm," pp. 653–660, Association for Computing Machinery, Inc, 7 2016.

[29] A. Szollos, M. Šmíd, and J. Hájek, "Aerodynamic optimization via multi-objective micro-genetic algorithm with range adaptation, knowledge-based reinitialization, crowding and ϵ-dominance," *Advances in Engineering Software*, vol. 40, pp. 419–430, 6 2009.

[30] R. Landa-Becerra, L. V. Santana-Quintero, and C. A. Coello, "Knowledge incorporation in multi-objective evolutionary algorithms," *Studies in Computational Intelligence*, vol. 98, pp. 23–46, 2008.

[31] K. Amouzgar, A. Nourmohammadi, and A. H. Ng, "Multi-objective optimisation of tool indexing problem: a mathematical model and a modified genetic algorithm," *International Journal of Production Research*, vol. 59, pp. 3572–3590, 2021.

[32] C. A. B. Diaz, T. Aslam, and A. H. Ng, "Optimizing reconfigurable manufacturing systems for fluctuating production volumes: A simulation-based multi-objective approach," *IEEE Access*, 2021.

[33] A. Nourmohammadi, H. Eskandari, M. Fathi, and A. H. Ng, "Integrated locating in-house logistics areas and transport vehicles selection problem in assembly lines," *International Journal of Production Research*, vol. 59, pp. 598–616, 2021.

[34] K. Amouzgar, A. H. Ng, and G. Ljustina, "Optimizing index positions on cnc tool magazines considering cutting tool life and duplicates," vol. 93, pp. 1508–1513, Elsevier, 1 2020.

[35] A. Jaszkiewicz and P. Zielniewicz, "Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem," *European Journal of Operational Research*, vol. 193, pp. 885–890, 3 2009.

[36] B. Christian and S. Cremaschi, "Planning pharmaceutical clinical trials under outcome uncertainty," 1 2018.

[37] A. Baghdadi, M. Heristchian, and H. Kloft, "Design of prefabricated wall-floor building systems using meta-heuristic optimization algorithms," *Automation in Construction*, vol. 114, p. 103156, 6 2020.

[38] H. Li, "Network topology design," 1 2016.

[39] S. K. Seo, D. H. Cho, Y. Lim, and C. J. Lee, "Application of genetic algorithm to layer patterning of plate fin heat exchanger," 10 2017.

[40] T. Rios, B. V. Stein, T. Back, B. Sendhoff, and S. Menzel, "Point2ffd: Learning shape representations of simulation-ready 3d models for engineering design optimization," pp. 1024–1033, Institute of Electrical and Electronics Engineers (IEEE), 1 2022.

[41] S. Friess, P. Tino, Z. Xu, S. Menzel, B. Sendhoff, and X. Yao, "Artificial neural networks as feature extractors in continuous evolutionary optimization," vol. 2021-July, Institute of Electrical and Electronics Engineers Inc., 7 2021.

[42] S. Friess, P. Tino, S. Menzel, B. Sendhoff, and X. Yao, "Representing experience in continuous evolutionary optimisation through problem-tailored search operators," Institute of Electrical and Electronics Engineers Inc., 7 2020.

[43] G. Ruan, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao, "Computational study on effectiveness of knowledge transfer in dynamic multi-objective optimization," Institute of Electrical and Electronics Engineers Inc., 7 2020.

[44] C. A. C. Coello and M. G. C. Tapia, "Cultural algorithms for optimization," 7 2021.

[45] S. Obayashi and D. Sasaki, “Visualization and data mining of pareto solutions using self-organizing map,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2632, pp. 796–809, 2003.

[46] S. Saha, T. Rios, L. L. Minku, X. Yao, Z. Xu, B. Sendhoff, and S. Menzel, “Optimal evolutionary optimization hyper-parameters to mimic human user behavior,” pp. 858–866, Institute of Electrical and Electronics Engineers Inc., 12 2019.

[47] I. Karlsson, S. Bandaru, and A. H. Ng, “Online knowledge extraction and preference guided multi-objective optimization in manufacturing,” *IEEE Access*, vol. 9, pp. 145382–145396, 2021.

[48] I. Morshedzadeh, A. H. Ng, and M. Jeusfeld, “Managing manufacturing data and information in product lifecycle management systems considering changes and revisions,” *International Journal of Product Lifecycle Management*, vol. 13, pp. 244–264, 2021.

[49] S. Lidberg, T. Aslam, L. Pehrsson, and A. H. Ng, “Optimizing real-world factory flows using aggregated discrete event simulation modelling: Creating decision-support through simulation-based optimization and knowledge-extraction,” *Flexible Services and Manufacturing Journal*, vol. 32, pp. 888–912, 12 2020.

[50] H. Smedberg, S. Bandaru, A. H. Ng, and K. Deb, “Trend mining 2.0: Automating the discovery of variable trends in the objective space,” Institute of Electrical and Electronics Engineers Inc., 7 2020.

[51] B. Xin, L. Chen, J. Chen, H. Ishibuchi, K. Hirota, and B. Liu, “Interactive multiobjective optimization: A review of the state-of-the-art,” 7 2018.

[52] K. Deb and J. Sundar, “Reference point based multi-objective optimization using evolutionary algorithms,” vol. 1, pp. 635–642, ACM Press, 2006.

[53] K. Miettinen, F. Ruiz, and A. P. Wierzbicki, “Introduction to multiobjective optimization: Interactive approaches,” 2008.

[54] L. Rachmawati and D. Srinivasan, “Incorporating the notion of relative importance of objectives in evolutionary multiobjective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 530–546, 8 2010.

[55] S. Greco, B. Matarazzo, and R. Słowiński, “Interactive evolutionary multiobjective optimization using dominance-based rough set approach,” 2010.

[56] C. Barba-González, V. Ojalehto, J. García-Nieto, A. J. Nebro, K. Miettinen, and J. F. Aldana-Montes, “Artificial decision maker driven by pso: An approach for testing reference point based interactive methods,” vol. 11101 LNCS, pp. 274–285, Springer Verlag, 2018.

[57] K. Deb, A. Sinha, P. J. Korhonen, and J. Wallenius, “An interactive evolutionary multiobjective optimization method based on progressively approximated value functions,” *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 723–739, 10 2010.

[58] M. Gombolay, R. Jensen, J. Stigile, T. Golen, N. Shah, S. H. Son, and J. Shah, "Human-machine collaborative optimization via apprenticeship scheduling," *Journal of Artificial Intelligence Research*, vol. 63, pp. 1–49, 5 2018.

[59] S. Greco, V. Mousseau, and R. Słowiński, "Ordinal regression revisited: Multiple criteria ranking using a set of additive value functions," *European Journal of Operational Research*, vol. 191, pp. 416–436, 12 2008.

[60] O. I. Larichev, "Cognitive validity in design of decision-aiding techniques," *Journal of Multi-Criteria Decision Analysis*, vol. 1, pp. 127–138, 1992.

[61] V. Ojalehto, D. Podkopaev, and K. Miettinen, "Towards automatic testing of reference point based interactive methods," vol. 9921 LNCS, pp. 483–492, Springer Verlag, 2016.

[62] S. Ruiyi, G. Liangjin, and F. Zijie, "Multi-objective collaborative optimization based on evolutionary algorithms," *Journal of Mechanical Design, Transactions of the ASME*, vol. 133, 10 2011.

[63] X. Shen, Y. Guo, Q. Chen, and W. Hu, "A multi-objective optimization evolutionary algorithm incorporating preference information based on fuzzy logic," *Computational Optimization and Applications*, vol. 46, pp. 159–188, 6 2010.

[64] J. Blank and K. Deb, "Solver: A blueprint for collaborative optimization in practice," vol. 1, pp. 105–110, 2021.

[65] S. Bandaru and K. Deb, "Automated innovization for simultaneous discovery of multiple rules in bi-objective problems," vol. 6576 LNCS, pp. 1–15, Springer, Berlin, Heidelberg, 2011.

[66] K. Deb, S. Bandaru, D. Greiner, A. Gaspar-Cunha, and C. C. Tutum, "An integrated approach to automated innovization for discovering useful design principles: Case studies from engineering," *Applied Soft Computing Journal*, vol. 15, pp. 42–56, 2 2014.

[67] K. Deb and A. Srinivasan, "Innovization: Discovery of innovative design principles through multiobjective evolutionary optimization," 11 2007.

[68] S. Bandaru, C. C. Tutum, K. Deb, and J. H. Hattel, "Higher-level innovization: A case study from friction stir welding process optimization," pp. 2782–2789, 2011.

[69] K. Deb, S. Gupta, D. Daum, J. Branke, A. K. Mall, and D. Padmanabhan, "Reliability-based optimization using evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 1054–1074, 2009.

[70] S. Bandaru and K. Deb, "Temporal innovization: Evolution of design principles using multi-objective optimization," vol. 9018, pp. 79–93, Springer Verlag, 2015.

[71] A. Gaur and K. Deb, "Towards an automated innovization method for handling discrete search spaces," pp. 2899–2906, Institute of Electrical and Electronics Engineers Inc., 9 2015.

[72] A. Gaur and K. Deb, “Adaptive use of innovization principles for a faster convergence of evolutionary multi-objective optimization algorithms,” pp. 75–76, Association for Computing Machinery, Inc, 7 2016.

[73] A. Gaur and K. Deb, “Effect of size and order of variables in rules for multi-objective repair-based innovization procedure,” pp. 2177–2184, Institute of Electrical and Electronics Engineers Inc., 7 2017.

[74] CPLEX, IBM ILOG, “V12. 1: User’s manual for CPLEX,” *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[75] Gurobi Optimization, LLC, “Gurobi Optimizer Reference Manual,” 2022.

[76] FICO, “Xpress-Optimizer Reference Manual,” 2009.

[77] ESTECO - ModeFRONTIER, “Process automation and optimization in the engineering designprocess,” 2022.

[78] V. R. D, “VisualDOC: Software for Process Integration and Multidiscipline Design Optimization,” 2022.

[79] Siemens, “HEEDS – Discover better designs, faster,” 2022.

[80] Noesis-Optimus, “The industry-leading pido software platform,” 2022.

[81] Ansys, “Ansys optiSLang - Process Integration & Design Optimization,” 2022.

[82] Dassault-iSight, “Automate design exploration and optimization,” 2022.

[83] J. J. Durillo and A. J. Nebro, “jmetal: A java framework for multi-objective optimization,” *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011.

[84] J. Durillo, A. Nebro, and E. Alba, “The jmetal framework for multi-objective optimization: Design and architecture,” in *CEC 2010*, (Barcelona, Spain), pp. 4138–4325, July 2010.

[85] J. Blank and K. Deb, “Pymoo: Multi-objective optimization in python,” *IEEE Access*, vol. 8, pp. 89497–89509, 2020.

[86] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, “PISA — a platform and programming language independent interface for search algorithms,” in *Evolutionary Multi-Criterion Optimization (EMO 2003)* (C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, eds.), Lecture Notes in Computer Science, (Berlin), pp. 494 – 508, Springer, 2003.

[87] K. Miettinen and M. M. Mäkelä, “Nimbus — interactive method for nondifferentiable multiobjective optimization problems,” pp. 50–57, 1996.

[88] G. Misitano, B. S. Saini, B. Afsar, B. Shavazipour, and K. Miettinen, “Desdeo: The modular and open source framework for interactive multiobjective optimization,” *IEEE Access*, vol. 9, pp. 148277–148295, 2021.

[89] Y. Dhebar and K. Deb, "Interpretable rule discovery through bilevel optimization of split-rules of nonlinear decision trees for classification problems," *IEEE Transactions on Cybernetics*, 2020.

[90] A. Ghosh, Y. Dhebar, R. Guha, K. Deb, S. Nageshrao, L. Zhu, E. Tseng, and D. Filev, "Interpretable AI agent through nonlinear decision trees for lane change problem," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 01–08, 2021.

[91] 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, 7 2020.

[92] A. Ghosh, K. Deb, R. Averill, and E. Goodman, "Combining user knowledge and online innovization for faster solution to multi-objective design optimization problems," vol. 12654 LNCS, pp. 102–114, Springer, Cham, 3 2021.

[93] A. Ghosh, K. Deb, E. Goodman, and R. Averill, "A user-guided innovization-based evolutionary algorithm framework for practical multi-objective optimization problems," *Engineering Optimization*, vol. 0, no. 0, pp. 1–13, 2022.

[94] 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," 2 2019.

[95] M. P. Bendsøe, O. Sigmund, M. P. Bendsøe, and O. Sigmund, "Topology design of truss structures," 2004.

[96] A. Ahrari, A. A. Atai, and K. Deb, "Simultaneous topology, shape and size optimization of truss structures by fully stressed design based on evolution strategy," *Engineering Optimization*, vol. 47, pp. 1063–1084, 8 2015.

[97] A. Ahrari, A. A. Atai, and K. Deb, "A customized bilevel optimization approach for solving large-scale truss design problems," *Engineering Optimization*, vol. 52, pp. 2062–2079, 12 2020.

[98] M. Z. A. Elrehim, M. A. Eid, and M. G. Sayed, "Structural optimization of concrete arch bridges using genetic algorithms," *Ain Shams Engineering Journal*, vol. 10, pp. 507–516, 2019.

[99] R. Cazacu and L. Grama, "Steel truss optimization using genetic algorithms and fea," *Procedia Technology*, vol. 12, pp. 339–346, 2014.

[100] 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, 3 2021.

[101] 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, 7 2018.

[102] 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, pp. 400–413, 6 2015.

[103] B. O. I. S. C. in Structural Optimization, "Optimization problem of iscso 2019," 2019.

[104] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms - a comparative case study," vol. 1498 LNCS, pp. 292–301, Springer Verlag, 1998.

[105] D. A. V. Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyses and new innovations," *IRE Transactions on Education*, vol. 8, pp. 125–147, 1999.

[106] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Modified distance calculation in generational distance and inverted generational distance," vol. 9019, pp. 110–125, Springer Verlag, 2015.

[107] J. D. Gibbons and S. Chakraborti, "Nonparametric statistical inference," 2011.

[108] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric Statistical Methods*. Wiley, 7 2015.

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

[110] A. V. Aho, M. R. Garey, and J. D. Ullman, "The transitive reduction of a directed graph," *SIAM Journal on Computing*, vol. 1, pp. 131–137, 7 1972.

[111] A. Rothwell, "Optimization of beams," 4 2017.

[112] S. Datta, A. Ghosh, K. Sanyal, and S. Das, "A radial boundary intersection aided interior point method for multi-objective optimization," *Information Sciences*, vol. 377, pp. 1–16, 2017.

[113] M. Basu, "Economic environmental dispatch using multi-objective differential evolution," *Applied Soft Computing*, vol. 11, pp. 2845–2853, 3 2011.

[114] S. Tan, S. Lin, L. Yang, A. Zhang, W. Shi, and H. Feng, "Multi-objective optimal power flow model for power system operation dispatching," IEEE Computer Society, 2013.

[115] M. Ghasemi, S. Ghavidel, M. M. Ghanbarian, M. Gharibzadeh, and A. A. Vahed, "Multi-objective optimal power flow considering the cost, emission, voltage deviation and power losses using multi-objective modified imperialist competitive algorithm," *Energy*, vol. 78, pp. 276–289, 12 2014.

[116] S. K. Dash and S. Mohanty, "Multi-objective economic emission load dispatch with nonlinear fuel cost and noninferior emission level functions for ieee-118 bus system," pp. 1371–1376, Institute of Electrical and Electronics Engineers Inc., 6 2015.

[117] M. Ghasemi, S. Ghavidel, M. M. Ghanbarian, and M. Gitizadeh, "Multi-objective optimal electric power planning in the power system using gaussian bare-bones imperialist competitive algorithm," *Information Sciences*, vol. 294, pp. 286–304, 2 2015.

[118] F. Tooryan, A. Ghosh, Y. Wang, S. Srivastava, E. Arvanitis, and V. D. Angelis, "Comprehensive design of a microgrid energy storage with guaranteed optimality," 2020.

[119] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, pp. 12–19, 2 2011.

[120] S. Hossain, "Visualization of bioinformatics data with dash bio," *PROC. OF THE 18th PYTHON IN SCIENCE CONF*, 2019.

[121] A. Ghosh, K. Deb, E. Goodman, and R. Averill, "An interactive knowledge-based multi-objective evolutionary algorithm framework for practical optimization problems," 2022.