ONLINE INNOVIZATION : TOWARDS KNOWLEDGE DISCOVERY AND ACHIEVING FASTER CONVERGENCE IN MULTI-OBJECTIVE OPTIMIZATION

By

Abhinav Gaur

A DISSERTATION

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

Electrical Engineering — Doctor of Philosophy

2020

ABSTRACT

ONLINE INNOVIZATION : TOWARDS KNOWLEDGE DISCOVERY AND ACHIEVING FASTER CONVERGENCE IN MULTI-OBJECTIVE OPTIMIZATION

By

Abhinav Gaur

"Innovization" is a task of learning common principles that exist among some or all of the Pareto-optimal solutions in a multi-objective optimization problem. Except a few earlier studies, most innovization related studies were performed on the final non-dominated solutions found by an evolutionary multi-objective algorithm either manually or by using a machine learning method. Recent studies have shown that these principles can be learned during intermediate iterations of an optimization run and simultaneously utilized in the same optimization run to repair variables to achieve a faster convergence to the Pareto-optimal set. This is what we are calling as "online innovization" as it is performed online during the run of an evolutionary multi-objective optimization algorithm. Special attention is paid to learning rules that are easier to interpret, such as short algebraic expressions, instead of complex decision trees or kernel based black box rules.

We begin by showing how to learn fixed form rules that are encountered frequently in multi-objective optimization problems. We also show how can we learn free form rules, that are linear combination of non-linear terms, using a custom genetic programming algorithm. We show how can we use the concept of 'knee' in PO set of solutions along with a custom dimensional penalty calculator to discard rules that may be overly complex, or inaccurate or just dimensionally incorrect. The results of rules learned using this custom genetic programming algorithm show that it is beneficial to let evolution learn the structure of rules while the constituent weights should be learned using some classical learning algorithm such as

linear regression or linear support vector machines. When the rules are implicit functions of the problem variables, we use a computationally inexpensive way of repairing the variables by turning the problem of repairing the variable into a single variable golden section search.

We show the proof of concept on test problems by learning fixed form rules among variables of the problem, which we then use during the same optimization run to repair variables. Different principles learned during an optimization run can involve different number of variables and/or variables that are common among a number of principles. Moreover, a preference order for repairing variables may play an important role for proper convergence. Thus, when multiple principles exist, it is important to use a strategy that is most beneficial for repairing evolving population of solutions.

The above methods are applied to a mix of test problems and engineering design problems. The results are encouraging and strongly supports the use of innovization task in enhancing the convergence of an evolutionary multi-objective optimization algorithms. Moreover, the custom genetic program developed in this work can be a useful machine learning tool for practitioners to learn human interpretable rules in the form of algebraic expressions.

Dedicated to Maa.

ACKNOWLEDGMENTS

I first want to thank my family for the bulwark of support they have provided me ever since I decided to leave my job and pursue a career in research eleven years ago. Without them, I could never have mustered so much courage in my decisions. A big thank you to my wife for supporting me with the voice of hope and courage at times when I found it hard to believe in myself. I owe her so many days in kitchen and house chores. I want to take a moment to thank my friends from my undergraduate days; Asjad, Akshat, Aman and Gandhi, who paid so many of my bills and university application fees when I left my job eleven years ago to pursue a career in research, always kept me in high spirits and even provided support to my parents in my absence from India.

Secondly, thanks to Prof. Kalyanmoy Deb for his invaluable time and advice throughout my doctoral studies. In terms of human qualities of intelligence, humility and fairness, he surely is a Utopian point. Quite literally, un-achievable, yet point of aspiration for all his students. I am very thankful to him for putting so much effort in arranging for my funding in every semester. I aspire to nurture this relationship in future and hopefully be part of growth of the COIN lab and Q-Lyst story in future.

These acknowledgements would not be complete without mentioning the immense support and camaraderie I received from my colleagues at COIN lab. Most notably Haitham, who selflessly helped me with so many discussions on optimization, Java programming and just life in general. I am so glad that he is my friend and now colleague at Ford Motor Company. Yashesh, who has been like a younger brother, whom I can even call in the middle of the night for help and who always made me feel more important than I should be probably. Rayan, whose smile in face of difficulties is enviable and I wish everyone could have that

fighting spirit that he possesses. Mohammad and Julian's hardwork is very inspiring and I absorbed some of that quality from them. Thanks to Proteek and Zhichao, who were always happy to discuss research problems and ideas. Not to forget Khaled, probably one of the best programmers that I have seen and with whom I learned so much about C++ programming in such a short time.

I want to thank General Motor's Vehicle Optimization Group and Manufacturing Analytics Group who at various points of time during my doctoral studies provided me with the opportunity to work on real world optimization problems and gave an invaluable experience of working on industry research problems. I want to acknowledge the BEACON's support for providing me with many generous travel grants to be able to attend international conferences and also the opportunity to meet and work with world class researchers of Evolutionary Computation all within a stones throw away from my lab.

I am sure there are more people who are part of this work in different ways and yet I am forgetting them. I want to assure them that this omission is not deliberate and a mere sign of my aging.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

KEY TO ABBREVIATIONS

CGP Customized Genetic Program

EA Evolutionary Algorithm

EMO Evolutionary Multi-objective Optimization

EMO/I Evolutionary Multi-objective Optimization with Innovization

GA Genetic Algorithm

GD Generational Distance

GP Genetic Programming

IGD Inverse Generational Distance

ML Machine Learning

MOO Multi-Objective Optimization

OLSR Ordinary Least Squares Regression

PO Pareto Optimal

SVM Support Vector Machines

Chapter 1

Introduction

These are exciting times for academicians, practitioners and entrepreneurs who are working in the field of Multi-objective optimization (MOO). Figure 1.1 shows the increasing attention this field has been receiving from academia over the last two decades. Commercial MOO software, especially catering to the computer aided engineering industry, has seen a lot of growth in the past decade. Now, practitioners have many options available for commercial MOO software; HEEDS [1], modeFrontier [2], , iSight [3], Optimus [4] and optiSLang [5] are to name a few. Figure 1.2 shows the rising use of these softwares in research. Evolutionary algorithms (EA) are a popular method of solving MOO problems in practice. In this work, we refer to EAs designed to solve MOO problems as *evolutionary multi-objective optimization* (EMO) algorithms. The key reason for the wide spread adoption of EMOs in tackling MOO problems is their flexibility in handling real world problem complexities such as non-linearity, non-convexity, and non-differentiability [6].

As the adoption of MOO is increasing in the industry, so are the expectations of practitioners from MOO solving software in terms of what it can do apart from providing optimal or near-optimal solution(s). Two key areas where researchers can try to address those expectations are;

1. Offering some insight or knowledge about the problem, and

2. Using the aforementioned knowledge in making an EMO algorithm converge faster

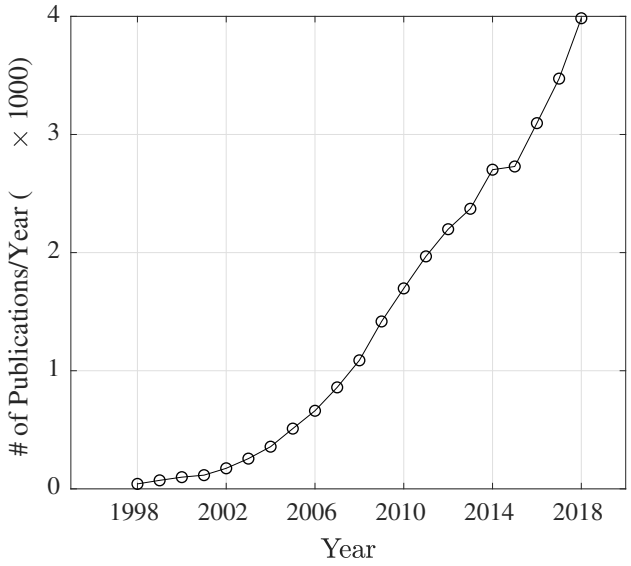


Figure 1.1: Number of publications in the last two decades using search key "multi-objective optimization" (Source : Scopus).

towards the PO front.

EMO algorithms or any population based MOO problem solving algorithm are uniquely positioned to deliver on both the aforementioned needs of MOO software users. Let us discuss the two aforementioned requirements in the context of this work.

1.1 Search for Knowledge in MOO Problems

The Oxford dictionary defines knowledge as, "the theoretical or practical understanding of a subject". In older Artificial Intelligence (AI) literature [7], "*having knowledge*" corresponds to *recognizing something as information* about the world or part of it. In the context of an engineering design and optimization problem, this is equivalent of understanding the optimal behaviour in relation to the objectives and the design variables of the problem. A closely related concept of "*knowledge representation*" is defined as fundamentally a surrogate, a substitute for the thing itself, used to enable an entity to determine consequences by

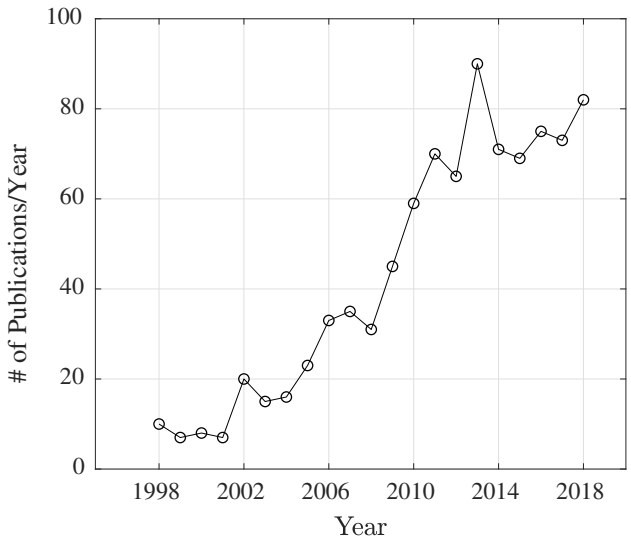


Figure 1.2: Number of publications in the last two decades that have used atleast one of the following MOO software-HEEDS, Mode-Frontier, iSight, Optimus, OptiSlang (Source : Scopus).

reasoning about the world rather than taking action in it [8]. Let us further discuss what we mean by *knowledge* in the context of this MOO problems. The MOO problems with two or more conflicting objectives have more than one optimal solution to the problem. These *pareto optimal* (PO) solutions may possess certain special properties that makes them PO in the first place. For example, in a mathematically well behaved MOO problem with convex, continuous and differentiable functions, the PO solutions must adhere to at least the *Fritz-John* or *Karush-Kuhn-Tucker* necessary conditions for Pareto optimality [9]. Even in real world engineering design problems that lack the aforementioned regularities around optima(s), the PO solutions may still possess certain special patterns or "*rules*" that sets them apart from non-optimal or even random solutions. This idea of looking for rules in PO solutions of MOO problem was coined by the authors of [10] as *innovization*. The authors showed the existence and manual extraction of such rules from PO solutions of many engineering design problems. Since then, the idea of innovization has been applied to many MOO problems in engineering design [11–13].

A recent work [14] provides an extensive survey of various forms of knowledge sought after in MOO problems. Knowledge representation in a MOO problem can be categorized into implicit or explicit forms based on representation [15]. Although very popular, knowledge in implicit form has no formal notation and may require user to have a specific experience. Most of the visual data mining methods such as parallel coordinate plots [16], value paths [17], heat-maps [18] and self-organizing maps [19] fall in this category. Explicit knowledge on the other hand has crisp mathematical form and can be interpreted by humans unambiguously. A couple of examples of explicit form of knowledge representation in MOO problems are decision trees [20] and regression rules [21]. In this work, we have decided to pursue knowledge in explicit form which has crisp mathematical notation and is amenable to consistent human "*interpretation*". Although we could not find a mathematical definition of interpretability, [22] defines it as the degree to which a human can consistently predict a model's result. For example, a model having power law structure in terms of design variables, which are common in the problems from engineering and physics [23], are easier to interpret compared to a model based on *deep neural networks* [24]. Christoph [25] presents a nice case for the need of interpretability in machine learning models. Figure 1.3 shows a sharp increase over last five years in the amount of research trying to address interpretability in machine learning. In this work, rules refer to explicit *mathematical expressions* involving MOO problem variables or functions thereof, that are adhered to by some or all of the PO solutions.

Another important aspect of rule learning in practical MOO problems is of adherence to principle of dimensional homogeneity [26]. In any rule representing some aspect of a physical or engineering system, adding or subtracting two dimensionally incommensurate quantities can never produce physically meaningful knowledge. For example, a rule should not be adding physical quantities having the dimensions of length and time. This is another

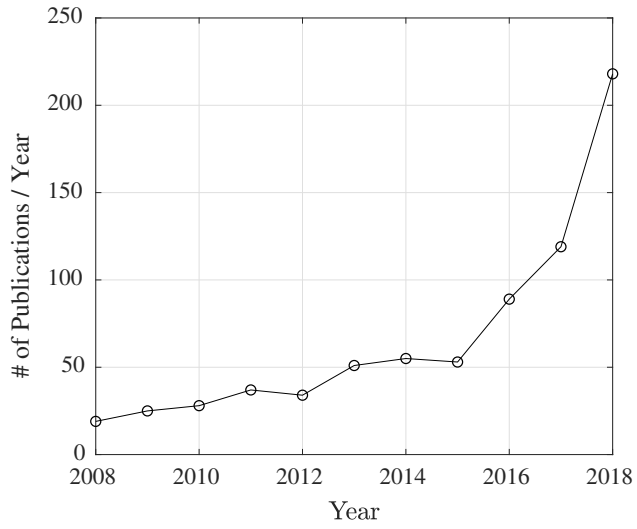


Figure 1.3: Number of publications per year in the last decade which have the keywords "machine learning" and "interpretable" in their title, abstract or keywords (Source : Scopus).

aspect that we have paid attention to while deciding the kind of rules that we want to learn from PO data. It is for these reasons of maintaining interpretability and easy verifiability of dimensional consistency, that we chose to learn only certain kind of rules in this work such as power law rules or rules involving basic operations of $\{+, -, \times, \div\}$ over problem variables. For the latter, we developed a bi-objective *genetic programming* (GP) [27] based method which we will go over in the coming chapters.

1.2 Using Discovered Knowledge in Expediting Convergence of EMO Algorithms

The EMO algorithms tend to be computationally inefficient because they evaluate many alternative solutions in the population of solutions before converging to PO solutions. There have been examples [28, 29] in which researchers have learned some heuristics about the problem from some initial EMO runs on a problem and then adopted those heuristics as

part of the problem solving EMO to expedite its convergence. This method is known as *derived heuristics* in the literature. In the above examples, the authors waited for the EMO algorithm run to completion, first to obtain a PO solutions set followed by a manual innovization procedure conducted over the solutions to come up with the rules. In this work, we intend to learn explicit mathematical form rules from a select set of solutions (say non-dominated set) during an EMO algorithm run and then use those rules to directly repair variables for expediting the convergence to PO solutions set. There are many challenging questions in this approach including;

- When to begin the learning process?
- What are some computationally efficient ways of repairing solution variables without making extra evaluations of the objective function?
- How to learn rules in MOO problems where different parts of PO front adhere to different set of rules?

We have tried to answer these questions towards the later half of this dissertation.

1.3 Organization of Dissertation

This dissertation is organized as follows. Chapter 2 presents relevant literature on the subject of innovization, existing literature on interpretable rule learning from data and on the idea of finding dimensionally consistent rules. Dimensional awareness here refers to adherence to the law of dimensional homogeneity by the learned rules. Then the dissertation is presented in two parts. Part I focuses on learning interpretable mathematical rules from data. Part I consists of four chapters. Chapter 3 illustrates how prevalent power laws are in engineering

problems and how can they be learned by using ordinary least square method of linear regresssion [30]. Chapters 4 and 6 show how can we learn free form rules use bi-objective GP for non-linear regression and classification tasks. We developed this GP as part of solving a real world industry problem. Chapter 5 then illustrates how can the principle of dimensional homogeneity be useful in generating or choosing physically meaningful rules from a set of rules which may all be acceptable to a user in terms of regression/classification accuracy. Part II then focuses on how can we use rules learned during an optimization task to expedite the convergence of an EMO algorithm. Chapter 7 shows results of online innovization on test and engineering MOO problems when we learn power law rules based on design variables only. Chapter 8 shows results of online innovization on test and engineering MOO problems when we learn power law rules based on design variables as well as some function of the same such as the objective functions. Furthermore, a methodology is devised to handle transition points in PO fronts and still be able to expedite the convergence of the EMO algorithm. We present the main conclusions of this work in Chapter 9 along with some interesting research threads worth pursuing in future research.

Chapter 2

Literature Survey and Proposed Online Innovization

2.1 Multi-objective Optimization

Many real-world design problems from engineering can be formulated as a Multi-objective optimization (MOO) problem [6, 31]. This approach is particularly useful when it is difficult to capture the decision makers' preference in the objectives in terms of some convex set of weights and consequently turning the problem into a single objective problem using the weighted sum approach . The goal of MOO is to provide the decision maker with a set of optimal trade-off solutions in terms of the objectives and let the decision maker develop a preference and finally make a choice. Equation (2.1) shows a formulation of a MOO problem with n_f objectives (minimize all), n_x design variables with known lower and upper bounds

and n_g inequality constraints.

$$\begin{aligned} \text{Minimize all in} \quad & \mathbf{f}(\mathbf{x}) \\ \text{Subect to} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ \text{where} \quad & \mathbf{x} = [x_1\ x_2\ \ldots\ x_{n_x}]^\intercal, \\ & \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x})\ f_2(\mathbf{x})\ \ldots\ f_{n_f}(\mathbf{x})]^\intercal, \\ & \mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x})\ g_2(\mathbf{x})\ \ldots\ g_{n_g}(\mathbf{x})]^\intercal \text{ and} \\ & l_i \leq x_i \leq u_i,\ i \in \{1, 2, \ldots, n_x\}. \end{aligned} \tag{2.1}$$

A *solution* $\mathbf{x}^* = [x_1^*\ x_2^*\ \ldots\ x_{n_x}^*]^\intercal$ to such a problem is a vector of n decision variables and must satisfy the variable bounds and n_g inequality constraints. Otherwise, it is an *infeasible* solution. Figure 2.1 illustrates the objective, design and feasible spaces for a bi-objective

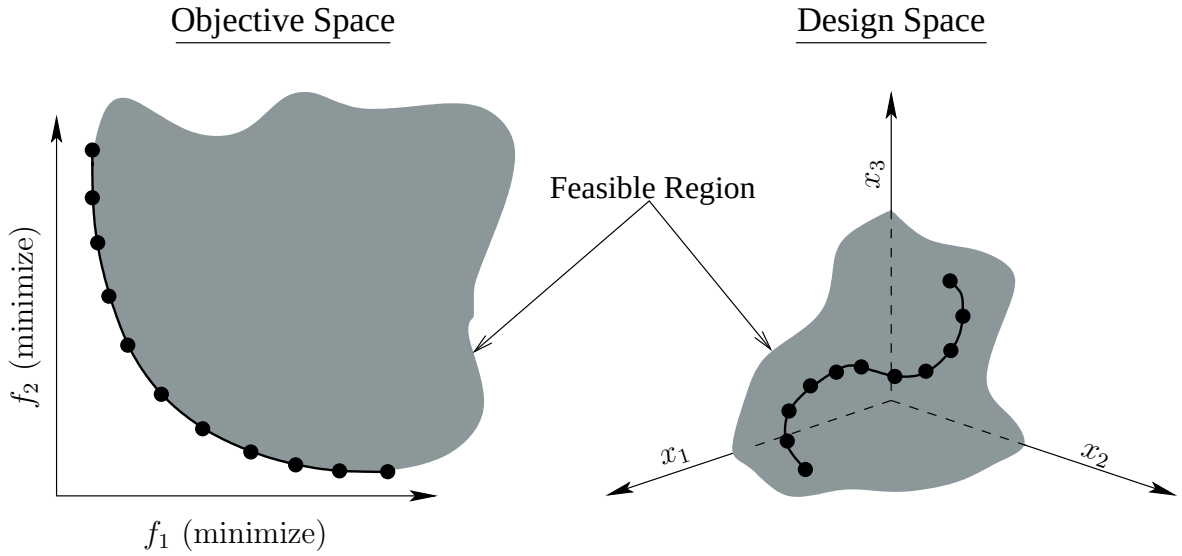


Figure 2.1: Illustration of the objective and design space of a MOO problem.

minimization problem having three design variables.

Concept of Dominance

In a single objective optimization problem, it is straight forward to compare two solutions simply based on their corresponding objective values. In case of minimizing an objective, the solution with lower objective value is considered better than the other. However, in case of MOO problem, the comparison is based on the concept of *dominance* [31].

Definition 2.1.1. A solution $\mathbf{x}^{(1)}$ is said to *dominate* a solution $\mathbf{x}^{(2)}$, if the following conditions are true:

1. The solution $\mathbf{x}^{(1)}$ is *no worse* than $\mathbf{x}^{(2)}$ in all objectives.

2. The solution $\mathbf{x}^{(1)}$ is *strictly better* than $\mathbf{x}^{(2)}$ in at least one objective.

We denote $\mathbf{x}^{(1)}$ dominating $\mathbf{x}^{(2)}$ as $\mathbf{x}^{(1)} \preceq \mathbf{x}^{(2)}$ and $\mathbf{x}^{(2)}$ dominating $\mathbf{x}^{(1)}$ as $\mathbf{x}^{(2)} \preceq \mathbf{x}^{(1)}$. If neither of two solutions dominate each other, then we denote this condition as $\mathbf{x}^{(1)} \parallel \mathbf{x}^{(2)}$. In Figure 2.2, upon doing pairwise comparisons of solution $\mathbf{x}^{(2)}$ with all others, $\mathbf{x}^{(2)} \preceq \mathbf{x}^{(4)}$,

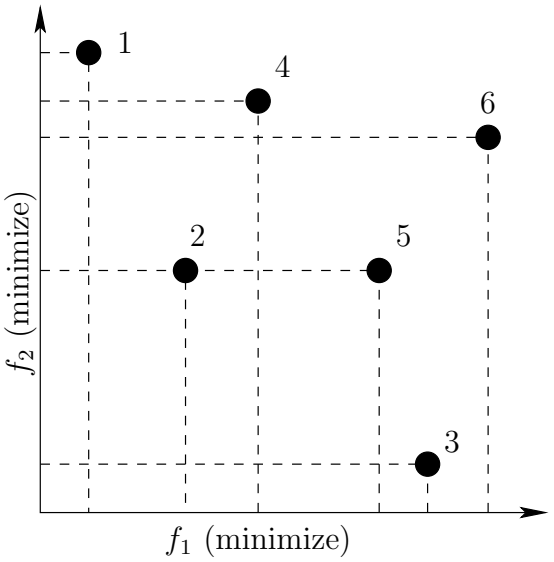


Figure 2.2: An illustration of the concept of dominance.

$\mathbf{x}^{(2)} \preceq \mathbf{x}^{(5)}$ and $\mathbf{x}^{(2)} \preceq \mathbf{x}^{(6)}$ while $\mathbf{x}^{(2)} \parallel \mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)} \parallel \mathbf{x}^{(3)}$. Another important concept is

of the *non-dominated set* [31].

Definition 2.1.2. Among a set of solutions P, the *non-dominated set* of solutions P' are those that are not dominated by any member of the set P.

In Figure 2.2, set $P = \{1, 2, 3, 4, 5, 6\}$ and set $P' = \{1, 2, 3\}$. Furthermore, when the set P is the entire search space, the resulting non-dominated set P' is called the *Pareto-optimal set*.

Sometimes, we are interested in dividing a set P into different non-domination levels or *ranks*. If a set P contains $J = \{1, 2, \ldots, j\}$ non-domination levels, then it means that the set P can be partitioned into j levels, i.e.

$$P = \bigcup_{i \in J} P_i$$

and

$$P_r \cap P_s = \varnothing \; \forall \; r \in J \text{ and } r \neq s,$$

such that a point in a set with lower non-domination rank are never dominated by a point in any set of higher non-domination rank. In Figure 2.2, there are three non-domination ranks present. Set $P_1 = \{1, 2, 3\}$, $P_2 = \{4, 5\}$ and $P_3 = \{6\}$.

2.2 Evolutionary Multi-objective Optimization Algorithms

Real world optimization problems have many complexities built into the problem such as non-linearity, discontinuity, non-differentiability, discrete variables etc. These complexities

make it impossible to apply known classical optimization algorithms, such as gradient descent algorithm [32], without making strong simplifying assumptions about the problem. An evolutionary algorithm (EA) mimics natural evolutionary principles to conduct search and optimization tasks. Some of the popular EAs are genetic algorithms [33], evolution strategies [34] and genetic programming [35]. Evolutionary algorithms (EAs) have shown to have an edge in optimizing MOO problem for finding multiple trade-off solutions, simply because EAs are capable of finding and storing multiple solutions from generations to generations. The population approach of EAs and their ability to build multiple niches within a population enabled EA researchers to develop *evolutionary multi-objective optimization* (EMO) algorithms [31,36]. EMO algorithms are extensively applied to MOO problems from the real world [37–44]. However, EMO algorithms are considered computationally inefficient because they sample the space stochastically and require many function evaluations to reach a high quality solution set. This inefficiency can be partly reduced if the algorithm is made to take advantage of some problem knowledge discovered during or at the end of the optimization from PO or non-dominated solutions set.

2.3 Innovization

While multiple PO solutions allow decision-makers to choose a single preferred solution by making a comparative analysis of them, multiple PO (or high-performing) solutions may also provide users with valuable information about the problem being solved. These PO solutions may possess certain special properties that makes them PO in the first place. For example, in a mathematically well behaved MOO problem with convex, continuous and differentiable functions, the PO solutions must adhere to at least the *Fritz-John* or *Karush-*

Kuhn-Tucker necessary conditions for Pareto optimality [9]. Even in real world engineering design problems that lack the aforementioned regularities around optima(s), the PO solutions may still possess certain special patterns or "*rules*" that sets them apart from non-optimal or even random solutions. The idea of learning from the PO solutions and deciphering design principles that are unique to the optimization problem was first presented in [10]. This concept is called *innovization* and it is pictorially illustrated in Figure 2.3. Originally, the innovization task was conceived to be performed once at the end of an EMO run in order to decipher principles that are common to most or all PO solutions [10].

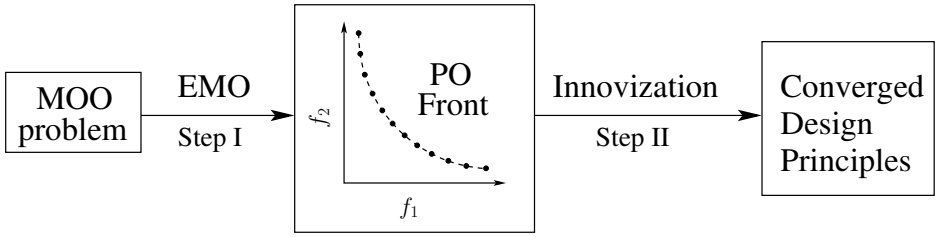


Figure 2.3: An illustration of the concept of innovization.

Since an innovization task involves learning from a set of equivalent trade-off solutions, it is appealing to couple this idea with EMO algorithms. There are examples in the literature, where the innovization task has been conducted on the results of an EMO algorithm to decipher interesting knowledge about the problem. For example, [45] uses innovization to discover new design principles in three engineering case studies. Another example [46] proposes a novel recommendation tool for software re-factoring based on innovization. [11] suggests a way to automate the finding of salient problem design principles for engineering problems. [13] uses the innovization task in a simulation based MOO problem and suggest different ways to couple the innovization with the EMO that can throw light on important principles of the problem. In recent literature [14], the idea of innovization has also been referred to as *knowledge discovery* in MOO problems. Before moving on to discuss some

examples of knowledge extracted from MOO problems, let us look at a taxonomy of the innovization procedures.

2.4 A Taxonomy of Innovization

In the following sections, we will try to cover the various kinds of innovization procedures that exist in the literature.

2.4.1 Manual Innovization

This was first proposed in [10] as a way of manually extracting innovative design principles from PO data of MOO design problems. As shown in Figure 2.3, it requires first solving a MOO problem and obtaining its PO set of solutions followed by manually plotting different variables, objectives and constraints for PO solutions against each other one by one in 2-dimensional and 3-dimensional plots and then manually deciphering innovative design knowledge from the same based on the plots.

2.4.2 Automated Innovization

Automated innovization was first proposed in [11] as a way to automate the labour of manual innovization but at the cost of fixing the form of rules to power law form. This approach has produced some interesting case studies in engineering design evident in [45]. Note that both manual and automated innovization are based on the approach shown in Figure 2.3.

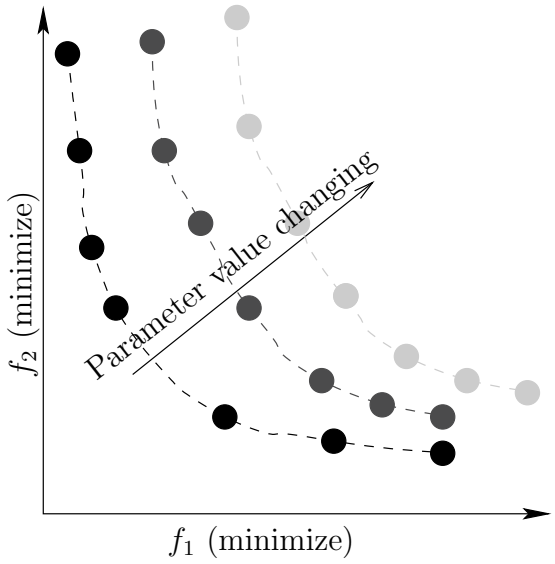


Figure 2.4: An illustration of the concept of higher level innovization.

2.4.3 Higher Level Innovization

Upon changing certain parameters of a MOO problem, it is possible that the shape or location or both may change for the PO front of the problem. Figure 2.4 illustrates this for a hypothetical problem where changing some parameter results in some shift in the PO front as well. In such a case, if an innovization study is performed for multiple fronts resulting from multiple parametric studies, it is possible that we may encounter certain rules or principles that remain invariant across the fronts. Such a procedure is called a higher level innovization procedure. Both [47, 48] present some good engineering design case studies of higher level innovization.

2.4.4 Lower Level Innovization

Lower level innovization task is performed when we can divide the solutions of a problem into two sets, namely the preferred set and the not preferred set. Figure 2.5 shows a hypothetical case where the user may be interested in patterns that are present in the preferred set of

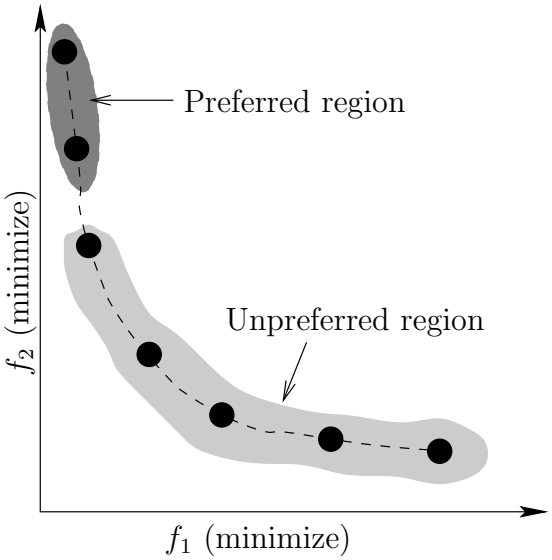


Figure 2.5: An illustration of the concept of lower level innovization.

solutions but not otherwise. A lower level innovization procedure learns the most discriminatory rules that completely adhered to by the preferred class of solutions but not so much by the not preferred class of solutions. [49] presents a case study of lower level innovization procedure applied to the Car side impact problem [50].

2.4.5 Temporal Innovization

Temporal innovization is the innovization procedure aimed at learning the relative hierarchy of design principles in an overall solution to a MOO problem. The design principles in the PO front are searched in all previous generations and their significance is recorded. When looked over the time line of all the generations of an EMO, it can be seen which principles evolve earlier in the solutions and which one appear later. The ones that appear earlier are hierarchically more important than the ones that appear later in the evolution of solutions. The authors of [51] present an interesting analogy between the evolution of design of a MEMS resonator using an EMO and the evolution of human beings.

2.5 Proposed Online Innovization

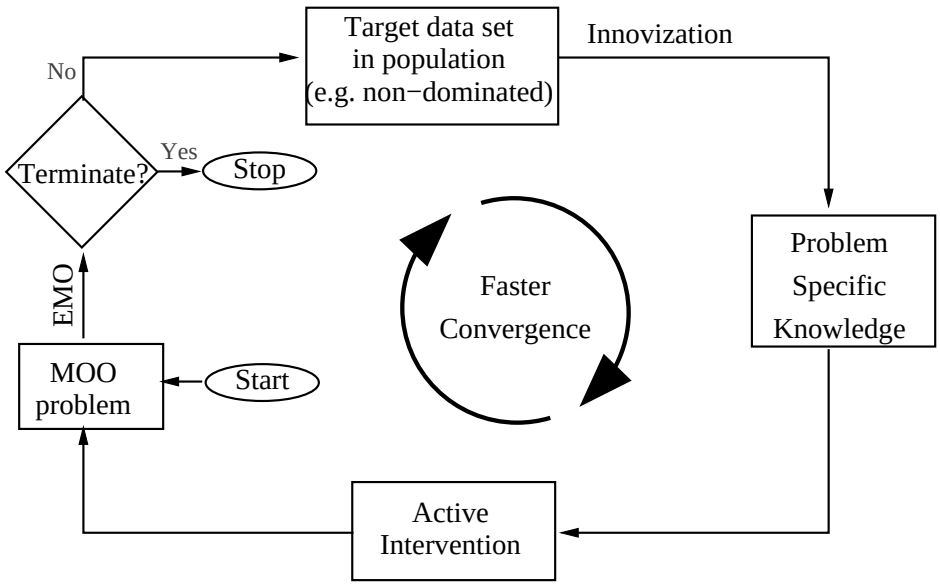


Figure 2.6: The concept of Online Innovization.

Figure 2.6 illustrates the idea of *online innovization*. An online innovization procedure has three components namely;

1. Target data set(s) for extracting knowledge,

2. Type of knowledge *representation* being extracted, and

3. How to use the extracted knowledge to expedite the convergence of EMO algorithm.

Let us look at these aspects.

2.5.1 Target Data Set(s)

When solving a MOO using an EMO or any population based optimization algorithm, the algorithm carries a number of data sets possessing important as well as unimportant knowledge about the problem. For example, the objective values data, variable values data and

constraint violation values data for non-dominated solutions set [31] can be considered important for problem information and the same data for dominated solutions may not hold the same importance, especially during initial generations of the EMO. However, the information contained the same dominated solutions becomes important if one is interested in finding knowledge that can discriminate between the dominated and non-dominated sets. Another way to select a target data set is if the preference of decision maker is known at every step of the optimization process [52].

2.5.2 Type of Knowledge Representation

The Oxford dictionary defines knowledge as, "the theoretical or practical understanding of a subject". In older Artificial Intelligence (AI) literature [7], "*having knowledge*" corresponds to *recognizing something as information* about the world or part of it. As mentioned in Section 1.1, "*knowledge representation*" is defined as fundamentally a surrogate, a substitute for the knowledge, used to enable an entity to determine consequences by reasoning about the world rather than taking action in it [8]. Knowledge representation in MOO problem can be categorized into implicit or explicit forms based on representation [15]. Although very popular, knowledge in implicit form has no formal notation and may require user to have a specific experience [53]. Most of the visual data mining methods used for extracting implicit knowledge about MOO problems such as parallel coordinate plots [16], value paths [17], heat-maps [18], RadViz visualization method [54] and self-organizing maps [19] fall in this category. [14] provides a good survey of such methods used for knowledge extraction in MOO problems.

Explicit knowledge on the other hand has crisp mathematical form and can be interpreted by humans unambiguously. Furthermore, it can be implemented as a computer program [55].

A couple of examples of explicit form of knowledge representation in MOO problems are decision trees [20] and regression rules [21]. Kernel based Support vector machines [56], artificial neural networks [57, 58] are other examples but rules learned through them are not very amenable to easy human interpretation. [59] applies classification rule mining [60] to extract rules of the form $A \rightarrow B$ where A is a design feature and B is a class-label that determines the quality of the design (e.g. non-dominated). However the rules in disjunctive normal form combining many design features and permutations thereof become difficult to interpret by human unless we limit the maximum number of features allowed in such rules. [11, 61] have developed a custom unsupervised learning algorithm based on a genetic algorithm to learn power law relations from the data of PO solutions of a MOO problem. This method does produce very interpretable rules in power law form, however it may fail to capture knowledge if it exists in any other form, say a rule with addition or substration operations. [62] employs classification trees to extract decision rules that distinguish between dominated and non-dominated solutions. These rules also stay easy to interpret only until the number of levels in the decision tree stays under five.

In this work, we have decided to pursue knowledge of the following two explicit forms:

- Power law form and
- Free form or algebraic expressions containing $\{+, -, \times, \div\}$ operations on operands.

We call these mathematical expressions as "*rules*" and target them as they are amenable to consistent human "*interpretation*". Although we could not find a mathematical definition of interpretability, [22] defines it as the degree to which a human can consistently predict a model's result. For example, a model having power law structure in terms of design variables, which are common in the problems from engineering and physics [23], are easier to interpret

compared to a more accurate model based on, say *deep neural networks* [24]. Christoph [25] presents a nice case for the need of interpretability in machine learning models. Although [11,61] have used a custom unsupervised learning algorithm to learn power law relations, such a method can be computationally very expensive, given that power laws can be learned using a combination of log-linear modeling along with ordinary least square regression method [63]. Such a method can be applied repeatedly without much cost, however we need to find the transition points in the MOO set via some other method. A transition point in a PO front is a point across which there is a quantitative change in the nature of rules that are adhered to by the PO solutions [10]. Encountering a constraint boundary is a common cause for appearance of transition points in PO fronts for MOO problems from engineering. Furthermore, we have also developed a custom bi-objective genetic programming (GP) method that can be used for learning simple "*free form* rules from data involving only $+, -, \times, \div$ operations. This is because adding rule parsimony as another objective in GP [64,65] has been known to control the problem of bloating [66] when using GP for regressing rules from data. Furthermore, we designed our GP to be able to learn the constants/coefficients involved in rules accurately without the constants being provided by the user. This capability is only known to be present in a commercial software named *Eureqa* [67] which is again a GP based software. For target rules involving only $+, -, \times, \div$ operations, our GP is quite competitive to Eureqa software. In addition to this, our GP has additional capability of conducting dimensional consistency checks on the obtained rules.

2.5.3 Using Extracted Knowledge to Expedite the Convergence

There are EAs that are hybridized with machine learning methods to acquire knowledge extraction from a set of solutions and knowledge application to affect the convergence. For

example Bayesian optimization [68, 69] and Estimation of Distribution Algorithms [70]. The probabilistic models constructed and updated in these algorithms guide the algorithm on how to sample the decision space to improve the objective values. However, the constructed probabilistic models are relatively difficult to interpret by humans compared to say algebraic expressions. Authors of [29] first discover the salient principles by using a manual innovization task with the solutions found at the end of an EMO run and then use those principles as a heuristic for local search and obtain a faster convergence than the EMO applied alone. [71] suggest learning the 'innovized' rules using decision trees and then adding the learned rules as if-then-else statement type of constraints during the EMO run. [71] learns distance based regression trees to extract rules differentiating between solutions near and far from PO region of interest and introduces these regression trees as constraints to guide the EMO algorithm towards the region of interest. Learnable evolution models (LEM) [72] generates multiple logical rules that relate specific design features to the high-quality solutions in the population. The rules are combined into one logical sentence in a disjunctive normal form, and future solutions must satisfy the sentence by conforming to at least one of the rules. [59] uses an adaptive operator selection to track the efficacy of each evolutionary operator and manually introduced knowledge dependent operator at consistently creating improved solutions and focus on using the most effective ones using a credit assignment strategy.

Since, we are learning rules in the form of mathematical expressions, we try to use the same for directly repairing/modifying solutions and possibly expedite the convergence of the EMO algorithm. Furthermore, we have tried to investigate different repair strategies for rule and repair-variable selection. We will learn more on this in Part-II of this dissertation.

2.6 Dimensional Awareness in Rule Learning

For rules to be physically meaningful, especially the ones coming from problems in engineering and physics, it is a must that the rules should at least adhere to law of dimensional homogeneity [26]. The first work to address the issue of learning dimensionally consistent rules from data using Genetic Programming (GP) was [73]. The authors modified the definitions of terminal set and function set to include the physical dimensions of the quantities, such as *length*, *mass* and *time* etc. Furthermore, an additional function `DimTransform()` is defined to guarantee closure. In addition to *goodness-of-fit* objective, an additional objective "*goodness of dimension*" is introduced to reduce the number of applications of the `DimTransform()` function necessary to make dimension based repairs. Figure 2.7 shows how the `DimTransform()` function is applied with an example. Here the input tree is adding two dimensionally inconsistent terms, namely "F" with physical dimensions $\mathtt{M}^1\ \mathtt{L}^1\ \mathtt{T}^{-2}$ and another quantity "m" with dimensions $\mathtt{M}^1\ \mathtt{L}^0\ \mathtt{T}^0$. The `DimTransform()` function multiplies the term "m" with a random physical constant "c" having physical dimensions of $\mathtt{M}^0\ \mathtt{L}^1\ \mathtt{T}^{-2}$ and a numeric value randomly chosen from the set of allowed random terminal constants. Consequent to this dimension matching operation, this transform adds a dimensional penalty of three units to the output tree. The work showed that addition of dimension information as another objective produced more parsimonious results instead of results having high fitness and low interpretability. Nevertheless, the algorithm did not perform well when exact scientific constants (e.g. 9.81 $\mathrm{m/s^2}$) were not provided to the algorithm.

[74] is another work that tries to address the issue of finding meaningful rules using dimension information using the idea of grammar based GPs [75, 76]. The authors do so by enforcing dimensional constraints through formal grammars in the GP tree construction

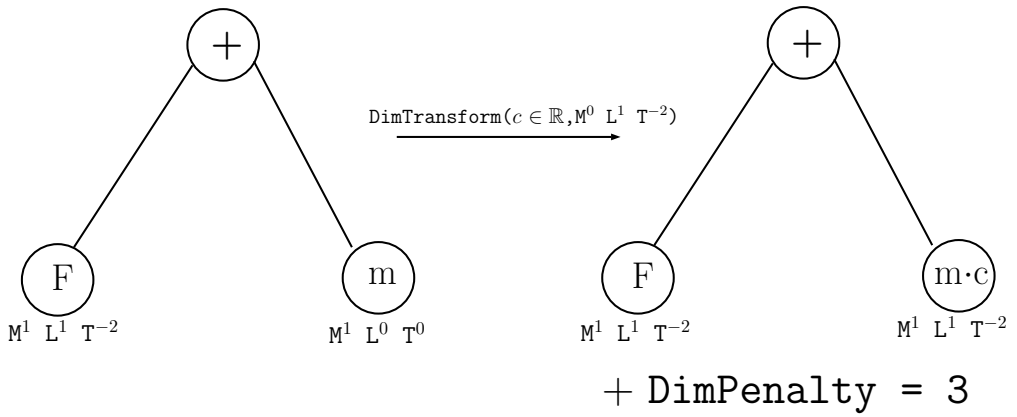


Figure 2.7: An example of `DimTransform()` function.

and manipulation rules. Furthermore, the authors also suggest a way of initialization so that feasible trees can be generated in pre-specified maximum tree depth. This approach is impractical for real problems because of the size of grammar that needs to be defined even for problems of small size. As an example, consider three elementary units *mass* (M), *length* (L) and *time* (T) to be present in the variables. Further assume that the powers of these basic units are restricted to the integer set $\{-2, -1, 0, 1, 2\}$ and no fractional powers of basic units are allowed. Thus one needs to exclude operators that yield fractional powers, e.g. *square root* operator. Using a restricted function set of $\{+, -, \div, \times\}$, a full specification of grammar requires $5^3 = 125$ symbols each having many rules. For example, for a symbol $S_{0,0,0}$ representing expressions which are dimensionless, can be obtained using multiplication

function using the following 125 rules.

$$
\begin{aligned}
< S_{0,0,0} > ::= & < S_{0,0,0} > \times < S_{0,0,0} > | \\
& < S_{1,0,0} > \times < S_{-1,0,0} > | \\
& < S_{2,0,0} > \times < S_{-2,0,0} > | \\
& \ldots \\
& [122 \text{ more such definitions for multiplication.}]
\end{aligned}
$$

Correctly so, the authors of [77] have termed this *context-free-grammar* approach to be unfit or impractical for modeling the "*units of measurement*" or *uom* system. It is very difficult to produce a set of feasible initial population with so many constraints, even with a limited set of allowed exponents.

The authors of [78] suggest using a set of dimensionless quantities as terminal set for the GP. From an initial set of variables, they generate a set of *normalized* variables which are dimensionless and then use those as part of the terminal set of the subsequent GP. This method, although guaranteed to produce dimensionally consistent expressions, is impractical when number of variables of interest is large. In such a case, finding a set of dimensionless quantities that can be produced using those variables of interest with elementary operations such as multiplication or division is not trivial and itself becomes an optimization problem.

The authors of [79] try addressing the dimensional consistency of GP individuals (trees) the following way. It first assumes an upper bound on the largest possible exponent for any fundamental unit (length, mass or time) to be u (say). Then for any tree, with maximum depth D, the maximum exponent of any fundamental quantity can be $u \times 2^D$. For each fundamental quantity, it then evaluates the resultant exponent which will exceed $u \times 2^D$

for trees combining dimensionally incommensurate quantities. Such trees become infeasible in subsequent optimization procedure to induce rules from data. However, such penalty approach makes it almost impossible to find rules having addition or subtraction operations, which are the only operations that make them different from power law rules. This is the reason that out of forty rules found by the author over three engineering design problems, only one rule has an addition operation.

In the rest of the chapters, we will focus on applying the idea of online innovization on MOO problems from the engineering design domain where the problem variables are of continuous nature. In the following chapters, we will see how can we:

- Part-I : learn mathematically explicit rules from data that are simple to interpret, and
- Part-II : then use them to make variable repairs to EMO population members to bring them closer to the PO front faster.

This concludes our literature survey chapter.

Part I

Rule Learning

Chapter 3

Learning Fixed Form Rules

3.1 Introduction

In a MOO problem being solved by any EMO or population based optimization algorithm generates a myriad of data related to the design variables, objective functions and constraint values/violations. Recall from Section 2.5.2 that we are interested in discovering knowledge from this data that can be expressed in the form of mathematical algebraic expressions which we will be referring to as rules in this work. Thus we are looking for rules of the form

$$\psi(\mathbf{x}, \mathbf{f}(\mathbf{x}), \mathbf{g}(\mathbf{x})) = c \tag{3.1}$$

where $\mathbf{x}$ is the set of design variables, $\mathbf{f}(\mathbf{x})$ are the objective functions and $\mathbf{g}(\mathbf{x})$ are the set of constraint functions for the problem and c is some constant. We can re-write Equation (3.1) as

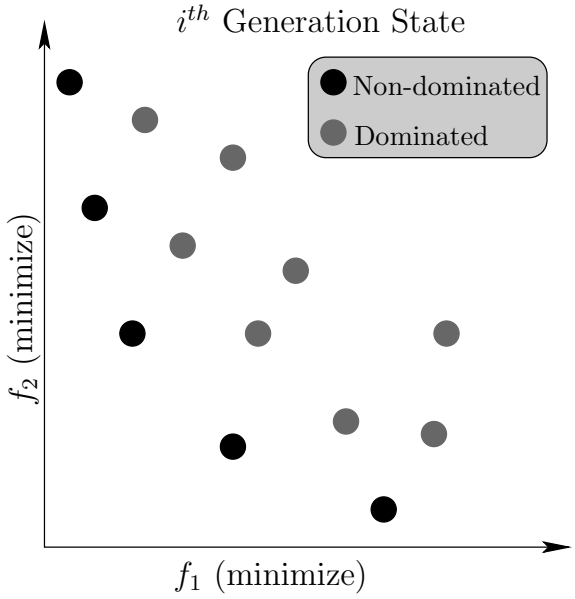
$$\psi(x_1, \ldots, x_{n_x}, f_1, \ldots, f_{n_f}, g_1, \ldots, g_{n_g}) = c. \tag{3.2}$$

Borrowing the terminology from [11], let us call each design variable and any function of design variables (including objective functions and constraint functions) as a basis function and we will represent a basis function using the symbol 'ϕ'.Thus, we can re-write Equation (3.2)

as

$$\psi(\phi_1, \phi_2, \ldots) = c. \tag{3.3}$$

Figure 3.1 shows a typical scenario of data availability while solving a MOO problem of



i^{th} Generation Data

ID	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	$\cdots$
	x_1	x_2	x_3	f_1	f_2	g_1	$\cdots$
1	0.2	0.8	0.8	4.5	2.8	1.2	$\cdots$
2	1.0	0.1	1.0	6.5	6.8	5.0	$\cdots$
3	1.0	0.4	0.7	7.1	6.6	9.6	$\cdots$
4	0.5	0.9	0.0	7.5	1.6	3.4	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Figure 3.1: Data availability when solving a MOO using an EMO algorithm.

two objectives, three variables and one inquality constraint. In some MOO problems, user may want to use additional basis functions other than the ones included in Equation (3.1), e.g. some composite function of first objective and a constraint. This is acceptable and would require us to calculate this additional basis function during the optimization run. We will focus our attention to the general case of including only variables, objectives and constraints for our study. An underlying assumption in applying these methods is that the values for each basis function are always non-negative, which is generally the case for MOO problems from engineering design domain. Note that the methodology to learn the two types of rules being presented in this chapter already exists in literature. We are presenting them for completeness, maintaining coherency of thesis structure and their suitability for

computationally efficient application to MOO engineering design problems.

3.2 Learning Constant Rules

We call a rule of the form

$$\psi(\phi_1, \phi_2, \ldots, \phi_{n_\phi}) \equiv \phi_i = c \tag{3.4}$$

as a "*constant*" rule where ϕ_i is some basis function from the set of n_ϕ basis functions being considered for rule learning from MOO problem and c is some constant. These are one of the most commonly encountered design rules in MOO problems in engineering designs. A few examples of a constant rule would be a variable or a constraint reaching its boundary value and being same for a set of high quality solutions.

3.2.1 Estimating Rule Parameters and Quality

In Equation (3.4), there is only one parameter, c to be estimated. Consider a hypothetical MOO problem which requires its PO solutions to have some variable $x_1 = 0.5$. Let x_1 be named the first basis function, ϕ_1. Figure 3.2 shows this scenario for this problem as the EMO algorithm progresses in the optimization run. Since EMO algorithms work based on stochastic sampling, a non-dominated front at the end of an EMO run is very likely not the PO front for the problem but an approximation of the same. EMO algorithms rarely reach the PO front of MOO problems unless they are followed by some local search from a set of non-dominated solutions which are close to the PO solutions of the problem. Hence, the value of x_1 cannot be expected to be exactly 0.5 for all solutions at the end. Figure 3.2 reflects this thought, where the values of the variable x_1 can be seen to be moving, on average, closer and closer to the value of 0.5 but are never exactly 0.5. Hence, the value of

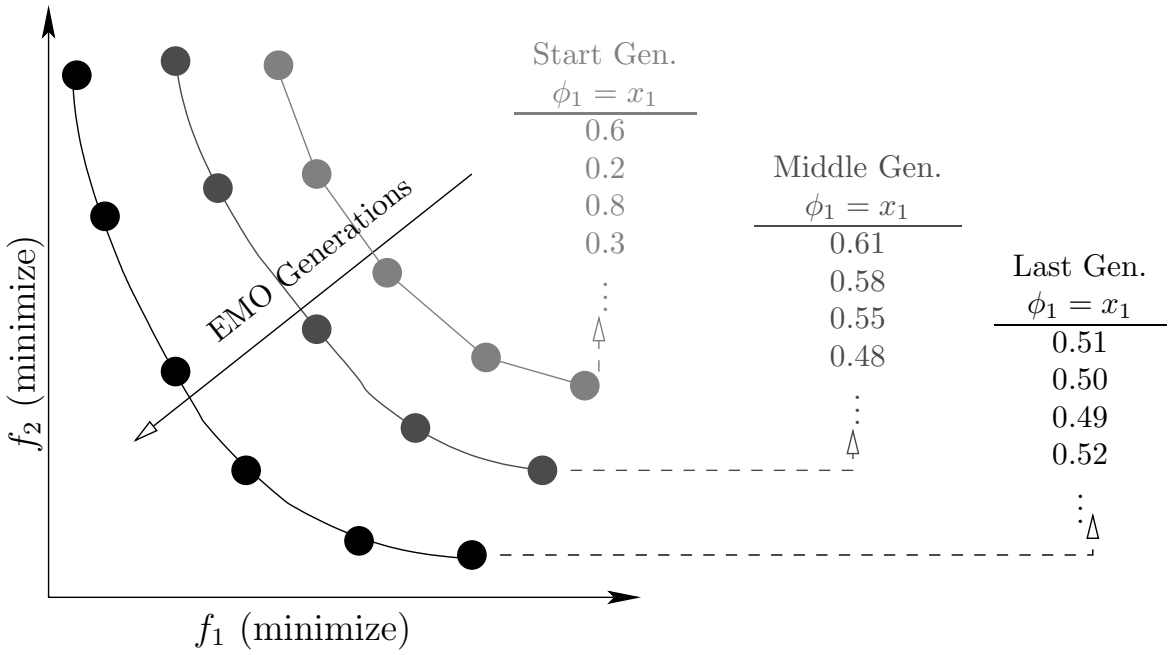


Figure 3.2: Data availability when solving a MOO using an EMO algorithm.

parameter c in Equation (3.4) can be estimated using the mean of basis function ϕ_i as

$$\hat{c} = \mu_{\phi_i} \tag{3.5}$$

which in this example would be just the mean of variable x_1 of the target data set.

Furthermore, we use *coefficient of variation* or $\mathcal{C}_v$ of ϕ_i values of target data set to assess the quality of a constant rule. Coefficient of variation for a data set is defined as

$$\mathcal{C}_v = \frac{\sigma}{\mu} \tag{3.6}$$

where σ is the standard deviation and μ is the mean of data. Coefficient of variation is a dimensionless quantity and is useful for comparing the variability of a sample of numbers drawn from distributions with different units. For example, $\mathcal{C}_v$ of the weights of a group of students will be the same whether the data is in kilograms or pounds. The lower the value

of $\mathcal{C}_v$ for data of some basis function, the closer it is to being called a constant rule. Let us now look at the other type of rule we are focusing on in this chapter, i.e. the power law rules.

3.3 Learning Power Law Rules

A power law form rule involving basis functions from a MOO can be represented as

$$\psi(\phi_1, \phi_2, \ldots, \phi_{n_\phi}) \equiv \prod_{i=1}^{n_\phi} \phi_i^{b_i} = c \tag{3.7}$$

where n_ϕ is the number of basis functions being considered for learning a power law rule from data, b_i is exponent of i^{th} basis function and c is some constant.

Equation (3.7) is a multi-variate power law. Such laws are very common in many natural phenomenon [23] and have been commonly found in MOO problems from engineering design [10, 11, 45, 80, 81]. By using such a universally occurring functional form for the design principles, it is expected that most correlations between various basis functions can be captured.

3.3.1 Estimating Parameters

Given the data for n_ϕ number of basis functions, and if we want to learn a multi-variate power law relation involving all k basis functions, then we can use the log-linear modeling followed by *ordinary least square regression* [82] method.

$$\phi_1^{b_1} \cdot \phi_2^{b_2} \cdots \phi_k^{b_k} = c, \text{ then} \tag{3.8}$$

taking log on both sides,

$$b_1 \log \phi_1 + b_2 \log \phi_2 + \cdots + b_k \log \phi_k = \log c, \tag{3.9}$$

which is a linear equation. If $\log \phi_1$ is chosen as the regressand and others as regressors then,

$$\begin{aligned} \log \phi_1 &= \frac{-b_2}{b_1} \log \phi_2 + \frac{-b_3}{b_1} \log \phi_3 + \cdots + \frac{-b_k}{b_1} \log \phi_k + \frac{\log c}{b_1}, \text{ or} \\ &= \hat{\beta}_2 \log \phi_2 + \hat{\beta}_3 \log \phi_3 + \cdots + \hat{\beta}_k \log \phi_k + \hat{\gamma}. \end{aligned} \tag{3.10}$$

Parameters $\hat{\beta}_2,\ \hat{\beta}_3, \cdots, \hat{\beta}_k$ and $\hat{\gamma}$ are estimates returned by a *Ordinary Least Square* linear regression (OLSR). OLSR also returns the R^2_{adj} value which can be used as metric to signify the quality of such a learned rule. Ofcourse we can choose any logarithmic term in Equation (3.9) as a regressand and the rest as regressors. Let us say that the i^{th} log term in Equation (3.9) is chosen as the regressand to apply OLSR. In that case, Equation (3.10) can be re-written as

$$\log \phi_i = \sum_{j=1, j \neq i}^{k} \left(\hat{\beta}_j \log \phi_j \right) + \hat{\gamma}, \tag{3.11}$$

which upon taking an anti-log on both sides can be re-written as

$$\phi_i = e^{\hat{\gamma}} \cdot \left(\prod_{j=1, j \neq i}^{k} \phi_j^{\hat{\beta}_j} \right). \tag{3.12}$$

3.3.2 Comparison with Automated Innovization

The authors of [11] developed an unsupervised machine learning method by combining grid based clustering [83] method with a genetic algorithm to learn rules of the form given by Equations (3.4) and (3.7). As part of automated innovization (see Section 2.4.2), authors

of [11] needed to learn such rules only once at the end of a MOO task using an EMO algorithm. However, in case of online innovization (see Section 2.5), we may have to learn such rules multiple times while, a MOO task is going on. If we use the method suggested in [11] to learn these rules, it will be computationally very inefficient because of an EA (for learning rules from MOO problem data) nested inside an EMO algorithm (for solving MOO problem).

Although our method is computationally much faster in obtaining power-law rules as compared to the method of automated innovization, however it holds a disadvantage as well. Our method cannot identify the rules that are applicable to only part of the PO front, say 50% of the PO front, and not the entire front. We will come back to this point in Part-7.2 where we have tried to address this shortcoming.

3.3.3 Learning Multiple Rules Simultaneously

It is possible that the non-dominated solutions of a MOO problem carry multiple design rules simultaneously. For example, not only some constraint g_1 is some constant upper bound but also two variables x_1 and x_2 are related in a power law form. Using the method given in Section 3.3, we can efficiently learn as many as our computational budget allows.

Consider a MOO problem shown in Equation (2.1) having n_f objectives, n_x design variables and n_g inequality constraints. Hence, if we are considering only variables, objective functions and constraint functions as basis functions, we have a minimum of $n_\phi = n_x + n_f + n_g$ basis functions for learning rules. With n_ϕ basis functions, exhaustively we have $2^{n_\phi} - 1$ number of constant and power-law rules for which we have to estimate the parameters and quality. These numerous rules for learning can become prohibitively large very quickly. Furthermore, the interpretability and use of such design rules for an engineer quickly diminishes

as the number of basis functions in a rule increases. For these reasons, we limit ourselves to learning rules with a maximum of two or three basis functions.

This chapter has covered the rule forms that are commonly encountered in MOO engineering design problems and their quality can also be ascertained quickly. However, we are also interested in learning rules where the form is not constrained a priori. In the next chapter, we will look at an efficient way of learning "*free-form*" non-linear rules from data.

Chapter 4

Learning Free Form Rules Using a Symbolic Regression Task

In Chapter 3, we saw how can we efficiently learn rules of two fixed forms, constant rules and power-law rules, in MOO data. These two forms are commonly encountered in MOO engineering design problems and can be learned quickly from data. However, in this chapter, we wish to expand our scope of knowledge extraction to rules with a weaker set of constraints limiting the form of the rules that we learn. Yet, we want to stay in the territory of mathematical/algebraic expressions that are simpler to interpret than, say some kernel based rules [84] and yet have the ability to capture complex physical behavior.

4.1 Form of Rules

We are focusing on learning rules of the form

$$\psi(\phi_1, \phi_2, \ldots, \phi_{n_\phi}) \equiv \phi_i = w_0 + \sum_{j=1, j\neq i}^{n_t} w_j \cdot t_j, \tag{4.1}$$

where ψ represents one such rule, ϕ are the basis functions explained in Section 3.1 that represent the data from MOO problem, ϕ_i is one of the basis functions that can either be a regressand in a symbolic regression problem or a class label in a classification problem, w_0

and w_js are numerical weights, n_t is the number of terms in the rule where each term t_j is some function of $\phi_j,\ j \in \{1, 2, \ldots, n_\phi\} \setminus i$ using the operations $\{+, -, \times, \div\}$. Such rules are a good candidate for learning using genetic programming [35].

4.2 A Primer on Genetic Programming

A GP can be considered an application of GAs when the space of solutions to search consists of programs or equations for solving a task [27, 35, 85]. Instead of decision variables, an individual is a program or an equation to solve a task. In order to create an initial population, a *terminal* set $\mathcal{T}$ and a *function* set $\mathcal{F}$ must be pre-specified. A terminal set consists of operands (constants and variables) where as a function set consists of operators or basic functions. Like other evolutionary computation techniques, a typical GP starts with a population of randomly created individuals, which in this case are math expressions. The fitness for each individual is determined by evaluating the rules. High fitness individuals are selected to form the mating pool, on which primary genetic operations, namely crossover and mutation, are applied to create a new population of programs. The process is repeated until some termination criterion (like maximum number of generations) is met. An individual in a GP can be represented using different data structures such as string of words [86], or trees [87] or graphs [88]. We will be using the tree data structure to represent a rule, hence we will discuss a few important concepts in the context of tree-based GPs.

Consider a GP with terminal set $\mathcal{T} = \{1, 2, x\}$ and function set $\mathcal{F} = \{+, -, *\}$. Then, Figure 4.1 shows two candidate solutions that belong to the set of valid GP individuals for such a GP. Furthermore, *sub-tree swap crossover* [89] is a popular crossover mechanism used in tree based GPs. An example of the same is shown in Figure 4.2. A sub-tree to be

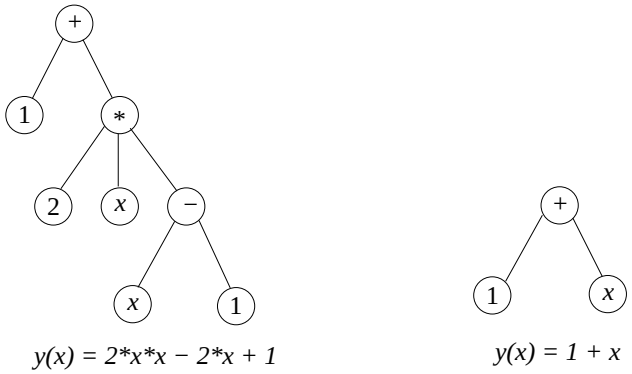


Figure 4.1: Example of two GP solutions using tree representation.

exchanged between two GP individuals (parents) is first chosen at random in each parent. Then, the sub-tree crossover operation is completed by exchanging the chosen sub-trees between the two parents. A Koza-style sub-tree mutation [35] involves swapping either a terminal with another element from the terminal set or a function with another element from the function set. Figure 4.3 shows an example of subtree mutation for an individual. When swapping functions, care must be taken to maintain the arity of functions being swapped are the same.

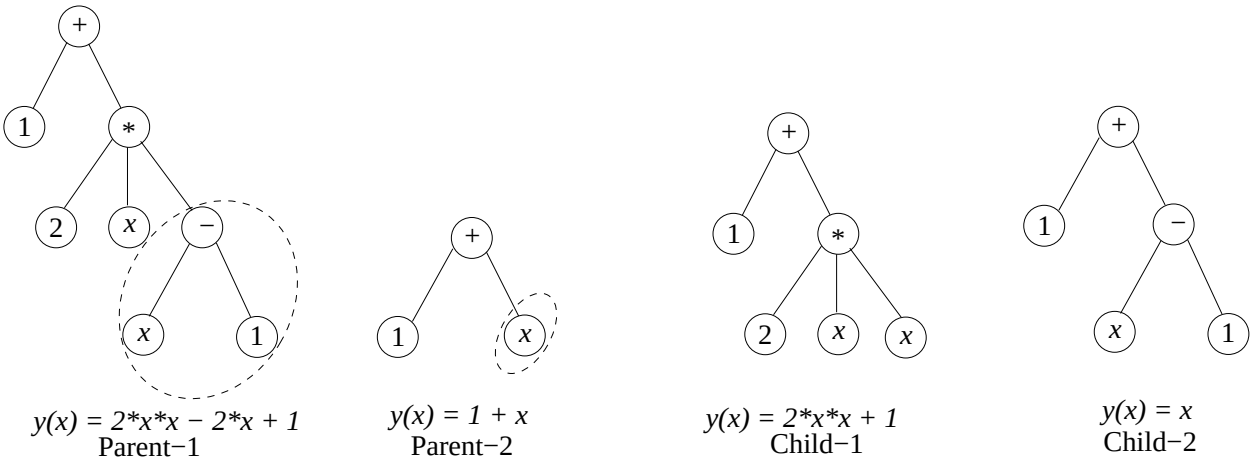


Figure 4.2: An example of a sub-tree crossover between two GP individuals.

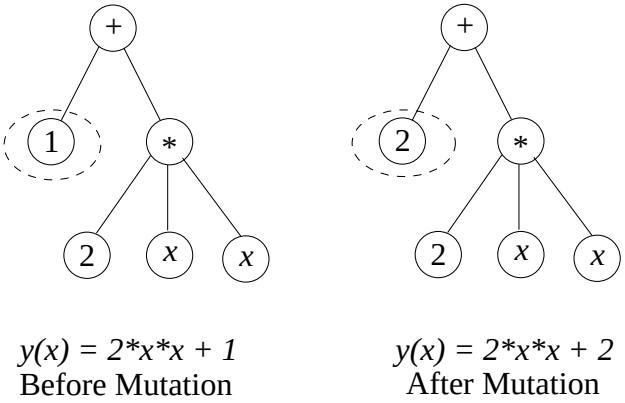


Figure 4.3: An example of a sub-tree Koza-style mutation of a GP individual.

4.3 A Custom GP (CGP)

GPs are used extensively at inducing mathematical models based on observations. Currently, two of the most popular commercial products for inducing mathematical models from data, Eureqa [67] and DataModeler [90], are GP based products. We developed our own *customized GP* or CGP by experimenting and combining with many different ideas from literature as well as our own to improve its performance. Some of the ideas exist in literature, however they have not been tried all together in the same GP implementation. This section describes the important ideas implemented in CGP along with its flowchart description towards the end.

4.3.1 Two Objectives : Prediction Error and Rule Complexity

Designer of any classification or regression technique faces this dilemma, i.e. whether to learn rules that are accurate or the ones that are simple for human interpretation [91]. Prediction error and rule *complexity* are conflicting objectives and we usually decide (unknowingly) apriori which one is more important for us by choosing a method. For a given problem, it is critical to have a clear idea of the which is a priority, accuracy or interpretability so that

this trade-off can be made explicitly rather than implicitly. Figure 4.4 shows this typical trade-off dilemma when selecting a machine learning method.

We decided to minimize both simultaneously by basing our GP on the bi-objective optimization algorithm NSGA-II [92]. Furthermore, having more than one objectives in a GP has known to reduce the problem of bloat [93]. Complexity of rule can be defined in many ways [94]. We define complexity of a rule as the number of nodes in a binary tree needed to represent the rule.

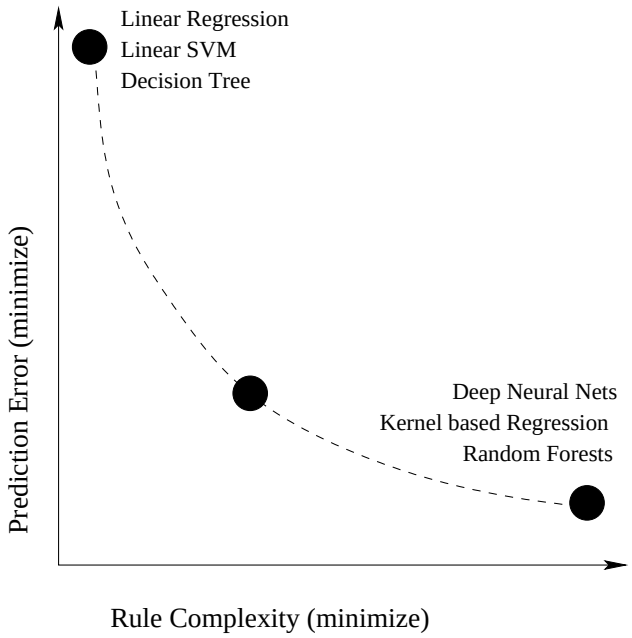
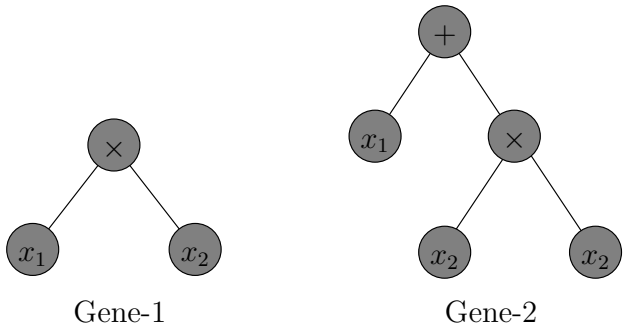


Figure 4.4: The two conflicting objectives in any machine learning algorithm.

4.3.2 Using Multiple Small Trees

As the complexity of rules to be learned increases, so does the size of its binary tree representation. This tree representation, which resides in the *phenotype* space of the GP is the object that undergoes important crossover and mutation operations. If the size of this tree is big, the possibility of a beneficial crossover reduces drastically. For this reason, [95]

suggested having multiple depth-limited small binary trees as phenotype of a GP individual instead of a single big tree and called the GP *multi-gene genetic program* or MGGP. Unlike traditional GP, each symbolic model (and each member of the GP population) in MGGP is a weighted linear combination of the outputs from a number of GP trees. Each of these trees may be considered to be a "gene". Figure 4.5 shows an example of a multi-gene GP individual. This fits very nicely with the type of rules shown in Equation (4.1) that we are aiming to learn.



$$y = w_0 + w_1 \cdot (x_1 \times x_2) + w_2 \cdot (x_1 + x_2^2)$$

Figure 4.5: An MGGP individual composed of many small binary expression trees instead of one big binary expression tree.

MGGP however treats the problem as a single objective optimization problem and faces the problem of *horizontal bloat* [95]. Horizontal bloating is the tendency of multi-gene models to acquire genes that are either performance neutral (i.e. deliver no improvement in prediction error on the training data) or offer very small incremental performance improvements. In case of CGP, having a bi-objective formulation helps allay this issue.

4.3.3 Learning Weights Using a Faster Method

Unlike traditional GPs that try to learn the constants/coefficients/weights in rule as well along with the rule structure, we separate that task completely from the GP. Figure 4.6

shows a sample individual with multiple genes. Like MGGP [95], we let evolution optimize the structure of the rule but use some gradient based optimization method, such as OLSR or linear *support vector machines*, to learn the weights in the rule.

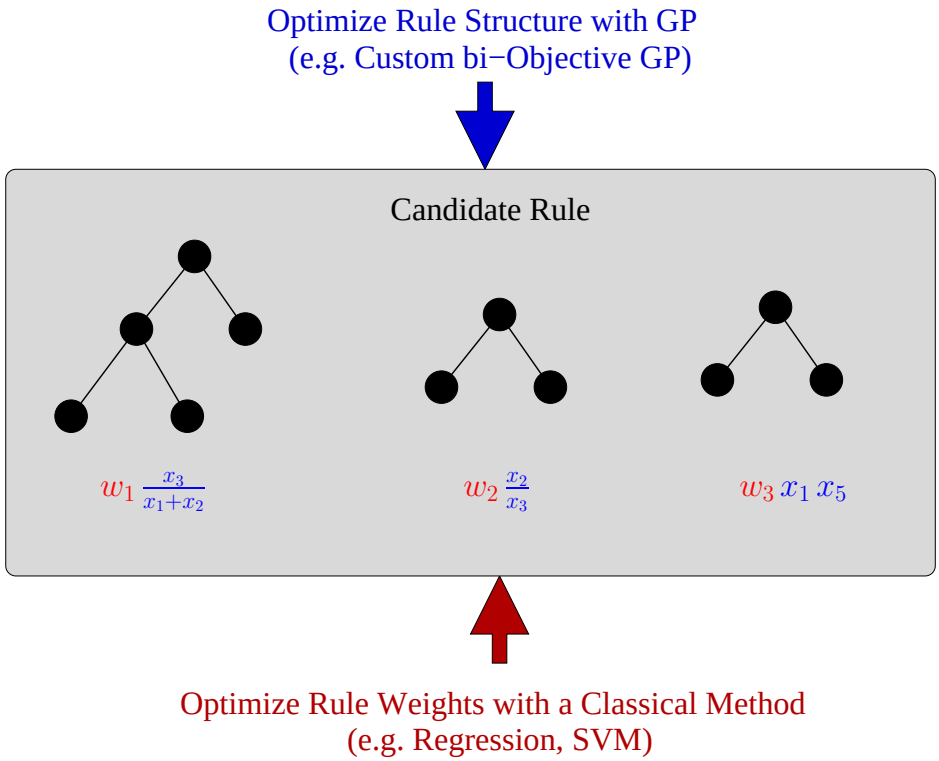


Figure 4.6: Our GP learns the rule structure and their weights separately.

4.3.4 Diversity Preserving Mechanism

In the bi-objective problem formulation mentioned in Section 4.3.1, the rule complexity objective is discrete in nature. This highly discretized objective space causes good individuals encountered early on in the CGP run to be reproduced and selected multiple times to produce many copies of these good solutions found early on. This leads to the population reducing to a few individuals and their duplicates, thus leading to complete loss in diversity and premature convergence of the algorithm [93].

To counter this, we need to adopt a strategy that can:

- Penalize all but one of each solution having duplicates (in both objectives) so that the ranking of the duplicate solutions worsens.

- The aforementioned penalization would push the non-domination rankings of duplicate solutions higher. However, the penalized duplicate(s) corresponding to a solution having a lower non-domination rank should continue to have lower non-domination rank compared to the penalized duplicate(s) corresponding to a solution having higher non-domination rank.

Figure 4.7 illustrates this idea with the help of a hypothetical example.

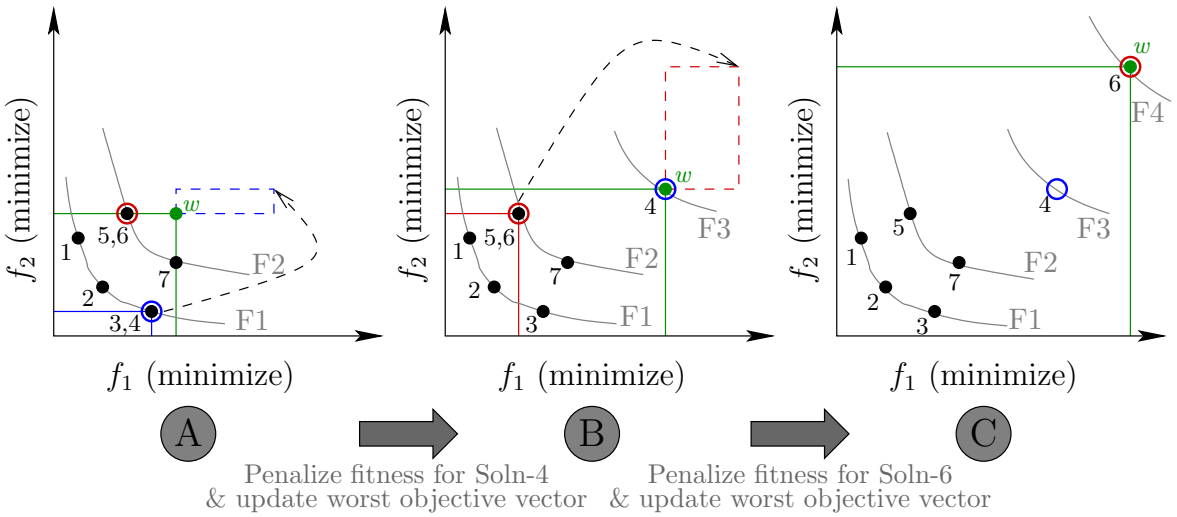


Figure 4.7: Fitness adjustment by penalizing duplicate solutions to promote population diversity in CGP. (A) shows the original state with two non-dominated fronts present in the population and total of seven solutions. (B) Solution-4, which is a duplicate of solution-3 after penalization. (C) Solution-6 which is a duplicate of solution-5 after penalization.

Part-A of Figure 4.7

- Part-A of Figure 4.7 shows the starting state of solutions. There are a total of seven solutions including two duplicate solutions.

- Solutions 1,2,3,5 and 7 are shown with black dots. Solution-4, which is a duplicate of solution-3, is shown with a blue circle. Similarly, solution-6, which is a duplicate of solution-5, is shown with a red circle.

- The green dot represents the worst objective vector [31].

- The non-dominated front F1 is comprised of solutions {1,2,3,4}. The non-dominated front F2 is comprised of solutions {5,6,7}.

- We begin by penalizing solution-4, the duplicate solution present in front F1. Let solution-4 be represented by objective vector $(f_1^{(4)}, f_2^{(4)})$ and the worst objective vector be represented as $(f_1^{(w)}, f_2^{(w)})$. Then the updated objective vector of solution-4 is given by

$$(f_1^{(4)}, f_2^{(4)}) \xleftarrow{assign} (f_1^{(4)} + f_1^{(w)}, f_2^{(4)} + f_2^{(w)}). \tag{4.2}$$

  This results in creating another non-dominated front, F3, for updated solution-4.

Part-B of Figure 4.7

- Part-B of Figure 4.7 shows the updated state of the solutions where updated solution-4 now belongs to a new non-dominated front F3 and the worst objective vector is updated as

$$(f_1^{(w)}, f_2^{(w)}) \xleftarrow{assign} (f_1^{(4)} + f_1^{(4)}). \tag{4.3}$$

- Next solution to be penalized is solution-6 (shown with red-circle) which is a duplicate of solution-5 (shown with a black dot). As in the case of solution-4, solution-6 is penalized as

$$(f_1^{(6)}, f_2^{(6)}) \xleftarrow{assign} (f_1^{(6)} + f_1^{(w)}, f_2^{(6)} + f_2^{(w)}). \tag{4.4}$$

This results in creating another non-dominated front, F4, for updated solution-6.

Part-C of Figure 4.7

- Part-C of Figure 4.7 shows the updated state of the solutions where penalized solution-4 now belongs to a new non-dominated front F3, penalized solution-6 now belongs to a new non-dominated front F4, and the worst objective vector is updated as

$$(f_1^{(w)}, f_2^{(w)}) \xleftarrow{assign} (f_1^{(6)} + f_1^{(6)}). \tag{4.5}$$

- Note that the relative non-domination ranking of solutions 4 and 6, both before and after penalizing, remains the same. Thus, the *non-domination hierarchy* of penalized solutions is preserved in this method.

- At this stage, no more duplicate solutions are present in any of the non-dominated fronts prompts the penalizing algorithm to stop.

This penalizing of duplicate solutions helps in reducing the clout of duplicates of good solutions and gives poorer solutions in higher non-dominated ranks a better chance at surviving and participating in parent selections of evolutionary algorithm.

4.3.5 Higher and Lower Level Crossovers

Recall from Section 4.3.2 that the genotypic space of CGP consists of many small trees or genes. Borrowing the idea from [95], CGP uses two types of crossovers namely low-level crossover and a high-level crossover. Any two parent individuals chosen to reproduce undergo a crossover with a probability p_c. With a (preferably) small probability when the

individuals do not go through a crossover operation, the outcome of the crossover operation are two child individuals that are identical copies of their parents. When crossover does happen, then it can either be of high-level type with a probability of p_{ch} or of low-level type with a probability $p_{cl} = 1 - p_{ch}$.

Consider two individuals from the CGP population, having three and two terms respectively as shown in the left half of Figure 4.8. Then for a high-level crossover to occur between these two individuals, CGP randomly chooses one term from each individual to cross and then swaps them between the individuals to create two children. This process is pictorially shown in Figure 4.8 where the second term of parent-1 is swapped with the second term of parent-2.

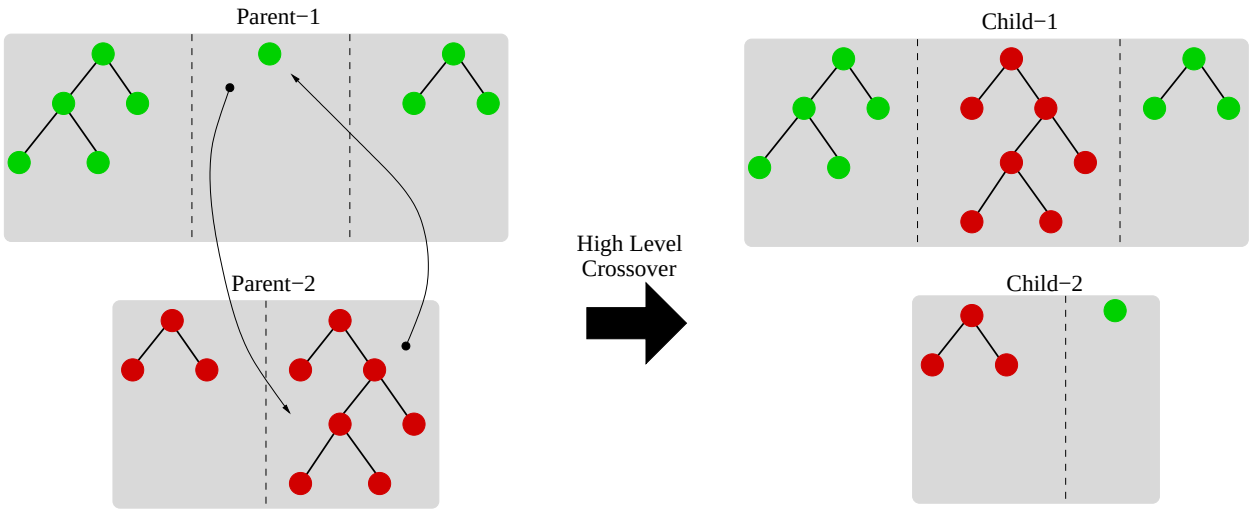


Figure 4.8: Example of the high-level crossover used in CGP.

If a low-level crossover need to be carried out, then CGP first chooses one term from each parent to cross and then carries out a sub-tree crossover among those two terms. This process is shown in Figure 4.9.

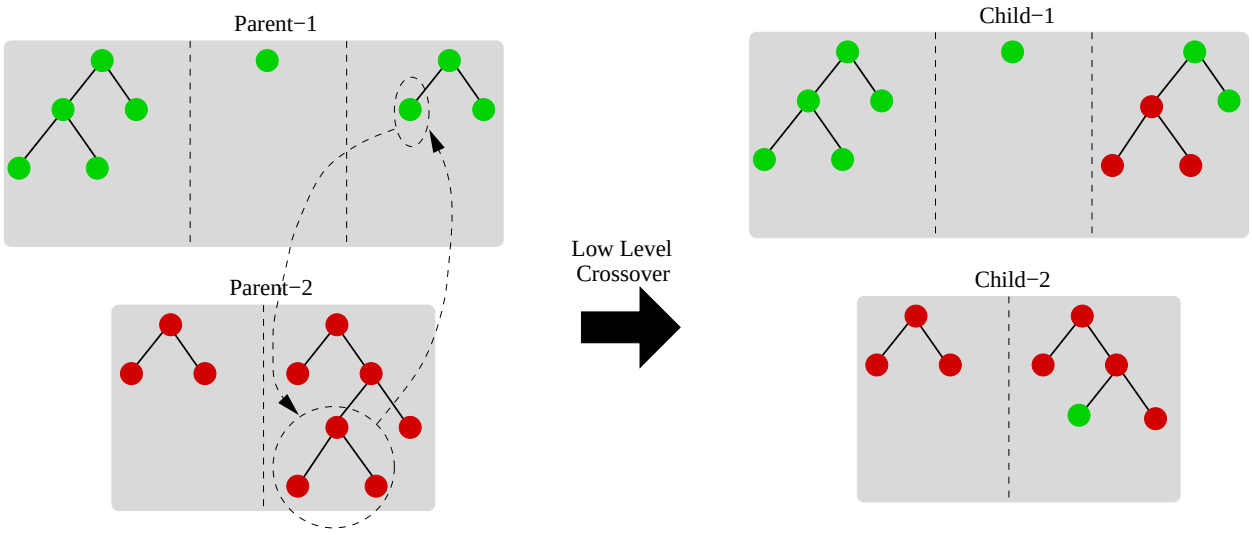


Figure 4.9: Example of the low-level crossover used in CGP.

4.3.6 CGP Flowchart for Rule Learning

Figure 4.10 shows the flowchart for the rule learning part of CGP. Except for one block, *penalizing duplicates*, everything else is same as that of NSGA-II algorithm. The algorithm begins with initialization of a population, say of N of individuals, composed of tree structures as explained in Section 4.3.2, each with not more than n_t trees. Each individual represents a rule of the form given in Eqation (4.1). The maximum depth [96] of each tree, say d_{max}, is also specified at time of initialization. Then the fitness functions are invoked to evaluate both prediction error on training set and complexity objectives for entire initial population. Then these individuals are assigned non-domination ranks and crowding distances [31] just like NSGA-II [92].

Once this parent population is ranked, the *parent selection* process produces list a of parents that are allowed to reproduce children for the next generation. CGP uses *binary tournament selection* [31] for selecting parents to reproduce. Such a parent selection process promotes the fittest individuals in the population to mate more often. Once these parents are selected, they go through genetic operations of crossover and mutation to produce a child

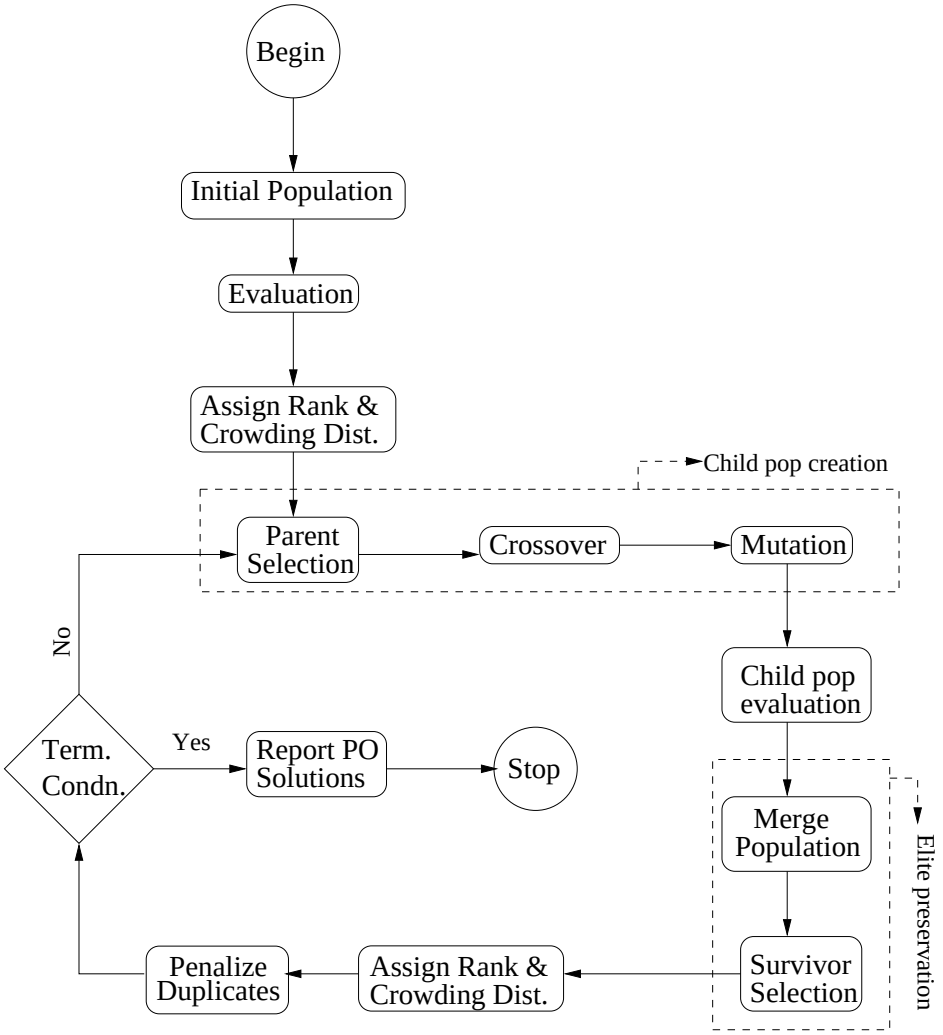


Figure 4.10: The flowchart for Custom Genetic Program or CGP.

population of N individuals. The crossover operation transpires via one of the two alternatives, either higher lever crossover or a lower-level crossover, both of which are explained in Section 4.3.5.

After the crossover operation, the N child individuals undergo mutation operation. For an individual, a mutation is carried out with probability p_m otherwise the child individual is left unchanged. In CGP, to mutate an individual (composed of many trees), first one of the trees is randomly selected for carrying out the mutation operation and then a Koza-style sub-tree mutation [35] is carried out on the chosen tree.

After undergoing the crossover and mutation operations, CGP evaluates the fitness of the N child individuals. Now these N children are combined with the N parent individuals of the current generation to obtain a merged population of size $2N$. This population of $2N$ individuals is passed on to the survivor selection procedure, where all the $2N$ individuals are again ranked and assigned crowding distances before selecting N individuals using the *crowded tournament selection operator* [31] used in NSGA-II.

This population of N individuals may then contain certain duplicate solutions. These duplicate solutions are penalized using the method given in Section 4.3.4 and the entire population is again assigned rank and crowding distance values. If termination condition is not met, these N individuals become the parent population for the next generation. This process goes on until the termination condition is met and the final PO set of solutions is reported. Table 4.1 shows the list of parameters for CGP.We now look at how can we use CGP for a symbolic regression task.

Table 4.1: List of parameters in CGP.

Symbol	**Description**	**Suggested Value**
N	Population size	50-200
G	Number of generations	100-500
n_t	Maximum number of terms in a CGP individual	3-10
$d_{\max}$	Maximum depth of trees representing individual terms of an individual	4-10
p_c	Probability of crossover	0.8-0.95
p_{ch}	Probability of high level crossover	0.2-0.3
p_{cl}	Probability of low level crossover	$1 - p_{ch}$
p_m	Probability of mutation	0.05-0.2

4.4 Using CGP for Symbolic Regression Task

Now that we are familiar with the working of CGP, recall that we wish to use CGP to learn rules of the form given by Equation (4.1) from MOO data as part of online innovization. This is a problem of symbolic regression and it is a type of regression analysis that searches the space of mathematical expressions to find the model that best fits a given data set. It is different and a difficult task as compared to conventional regression analysis which seeks to optimize the parameters for a pre-specified mathematical model structure involving regressand and regressors.

4.4.1 Evaluating Fitness of a CGP Individual for a Symbolic Regression Task

Recall from Section 4.3.1 that CGP approaches rule learning problem as a bi-objective optimization problem minimizing prediction error and rule complexity as the two objectives. Lets look at how these two objectives are calculated one by one.

The prediction error fitness function in CGP for symbolic regression task is an extension of *ordinary least squares regression* (OLSR) estimation method of linear regression. Consider a linear regression model

$$\boldsymbol{y} = \mathcal{Z}\boldsymbol{w} + \boldsymbol{\epsilon}, \tag{4.6}$$

where $\mathcal{Z} \in \mathbb{R}^{n\times(k+1)}$ is a matrix with n observations on k independent variables and the first column of $\mathcal{Z}$ contains only ones to include the bias term in the regression model, $\boldsymbol{y} \in \mathbb{R}^{n\times 1}$ is a vector of observations on the dependent variable, $\boldsymbol{\epsilon} \in \mathbb{R}^{n\times 1}$ is a vector of errors and $\boldsymbol{w} \in \mathbb{R}^{(k+1)\times 1}$ is a vector of unknown regression coefficients including the w_0 bias term that

we wish to estimate. Equation (4.6) in expanded form can be written as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & Z_{11} & \cdots & Z_{k1} \\ 1 & Z_{21} & \cdots & Z_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Z_{n1} & \cdots & Z_{kn} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}. \tag{4.7}$$

Then, the OLS estimate of the regression coefficients [97] $w_i,\ i \in \{0, 1, \ldots, k\}$ is given by

$$\hat{\boldsymbol{w}} = (\mathcal{Z}^\intercal \mathcal{Z})^{-1} \mathcal{Z}^\intercal \boldsymbol{y}. \tag{4.8}$$

Consider a five variable data set having n observations where we wish to symbolically regress a relation of the form $\boldsymbol{x}_4 = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_5)$. When we try to solve this problem using CGP , the CGP will initialize with a number of random CGP individuals. Consider an example of such an individual as shown in Figure 4.6 having three terms, $\boldsymbol{x}_3./(\boldsymbol{x}_1 + \boldsymbol{x}_2)$, $\boldsymbol{x}_2./\boldsymbol{x}_3$ and $\boldsymbol{x}_1. * \boldsymbol{x}_5$, where the operations './' and '.*' represent element wise division and multiplication of two vectors. For finding rules of the form shown in Equation (4.1) using this individual, the problem boils down to finding the appropriate weights $w_i,\ i \in \{0, 1, 2, 3\}$ in the relation

$$\begin{bmatrix} x_{41} \\ x_{42} \\ \vdots \\ x_{4n} \end{bmatrix} = \begin{bmatrix} 1 & \dfrac{x_{31}}{x_{11}+x_{21}} & \dfrac{x_{21}}{x_{31}} & x_{11}\cdot x_{51} \\ 1 & \dfrac{x_{32}}{x_{12}+x_{22}} & \dfrac{x_{22}}{x_{32}} & x_{12}\cdot x_{52} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \dfrac{x_{3n}}{x_{1n}+x_{2n}} & \dfrac{x_{2n}}{x_{3n}} & x_{1n}\cdot x_{5n} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}. \tag{4.9}$$

Comparing Equation (4.9) with (4.7), then we can obtain the weight estimates using Equation (4.8). The $\hat{\boldsymbol{x}}_4$ calculated for all observations can then be compared with the actual $\boldsymbol{x}_4$ values to get prediction error as $\boldsymbol{e} = \boldsymbol{x}_4 - \hat{\boldsymbol{x}}_4$. Note that $\boldsymbol{e} \in \mathbb{R}^{n\times 1}$ is a vector of prediction errors where n is the number of observations. Then for this individual, the fitness value or error objective, say f_{error}, can be calculated as a root mean square of e over all observations.

$$f_{\text{error}} = \sqrt{\frac{\boldsymbol{e}^\intercal \boldsymbol{e}}{n}} \tag{4.10}$$

The complexity fitness function simply calculates the total number of nodes in the trees of all terms of a CGP individual, i.e.

$$f_{\text{complexity}} = \sum(\text{ Nodes in all terms of CGP individual}). \tag{4.11}$$

Again, consider the example of a CGP individual shown in Figure 4.6 having three terms. The $f_{\text{complexity}} = 11$ for this individual.

4.5 CGP Results on Test Problems

Let us look at some results of symbolic regression on test problems using CGP .

4.5.1 Test Problem-1

This problem has a single variable x_1 as regressor for the regressand y. The data for this problem was generated using the equation

$$y = x_1 - \frac{x_1^3}{3!} + \frac{x_1^5}{5!} \tag{4.12}$$

where 100 values of x_1 are sampled from a uniform distribution over $[-\pi, \pi]$. Figure 4.11 shows the relation between x_1 and y graphically. The CGP parameters used in solving

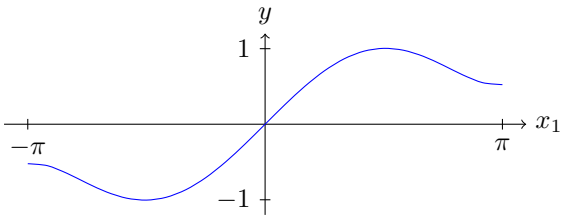


Figure 4.11: Graphical representation of test problem of Section 4.5.1.

this problem are given in Table 4.2. The PO solutions obtained by CGP for this problem are shown in Figure 4.12 in which each point represents a regressed non-linear relation between y and x_1. Also shown in Figure 4.12 is the regressed rule for a *knee* [98] solution. The trees corresponding to the three terms in this knee solution are shown in Figure 4.13. The expression for the chosen knee solution is almost same as the true relation shown in Equation (4.12) except for a small bias term of $1.155 \cdot 10^{-14}$ which is a numerical error and can be ignored.

Table 4.2: List of CGP parameters used for solving symbolic regression problem of Section 4.5.1.

Parameter	N	G	n_t	d_{max}	p_c	p_{ch}	p_{cl}	p_m
Value	20	30	10	10	0.95	0.20	0.80	0.05

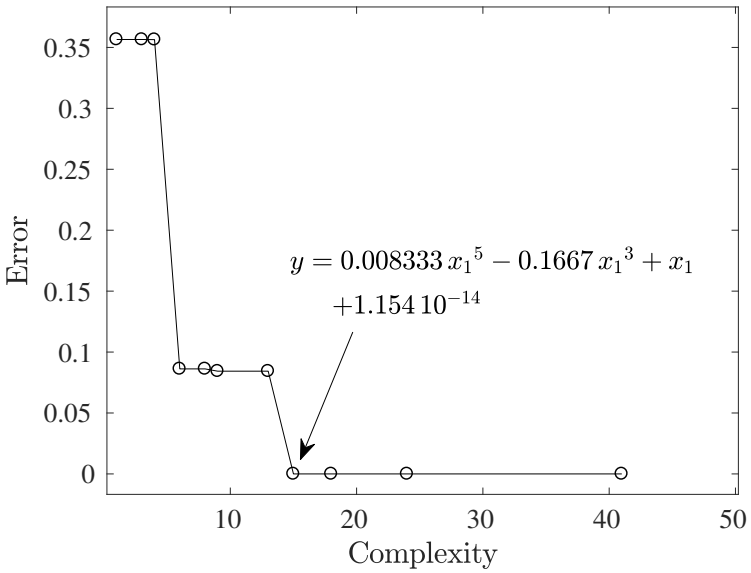


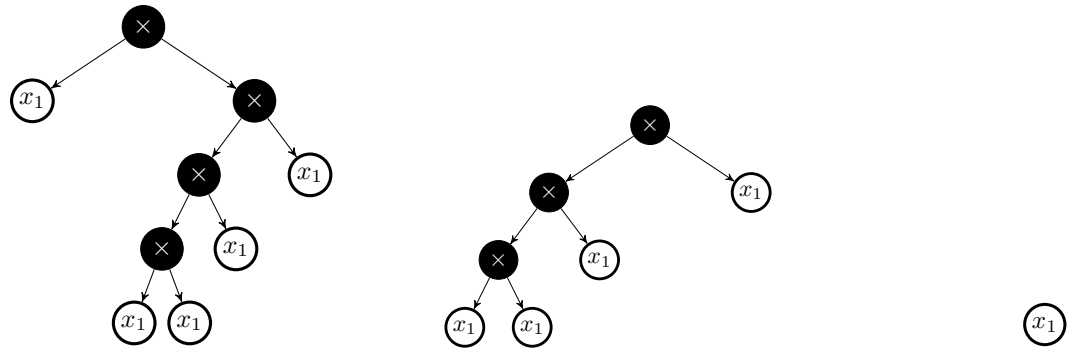
Figure 4.12: PO set of solutions found by CGP on solving the symbolic regression test problem of Section 4.5.1.

4.5.2 Test Problem-2

This problem has a single variable x_1 as regressor for the regressand y. The data for this problem was generated using the equation

$$y = \frac{x_1 + 1}{x_1^2 + x_1 + 1} \tag{4.13}$$

where 100 values of x_1 are sampled from a uniform distribution over $[-2, 2]$. Figure 4.14 shows the relation between x_1 and y graphically. The CGP parameters used in solving this problem are given in Table 4.3. The PO solutions obtained by CGP for this problem are shown in Figure 4.15 in which each point represents a regressed non-linear relation between y and x_1. Also shown in Figure 4.15 is the regressed rule for the solution with least error



(a) t_1 (b) t_2 (c) t_3

Figure 4.13: The three trees corresponding to the chosen knee solution shown in Figure 4.12.

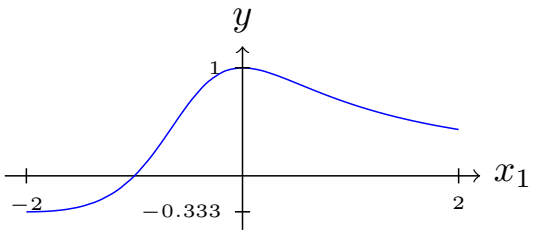


Figure 4.14: Graphical representation of test problem of Section 4.5.2.

and highest complexity among the PO solutions. The tree corresponding to this term is shown in Figure 4.16. The expression for the chosen solution is exactly the same as the true relation shown in Equation (4.13).

Table 4.3: List of CGP parameters used for solving symbolic regression problem of Section 4.5.2.

Parameter	N	G	n_t	d_{max}	p_c	p_{ch}	p_{cl}	p_m
Value	50	50	10	6	0.85	0.20	0.80	0.05

4.5.3 Test Problem-3

The *Bernoulli* equation [99] is a famous equation from the subject of fluid mechanics. Consider an incompressible fluid flowing under steady flow condition. Then, valid at any arbi-

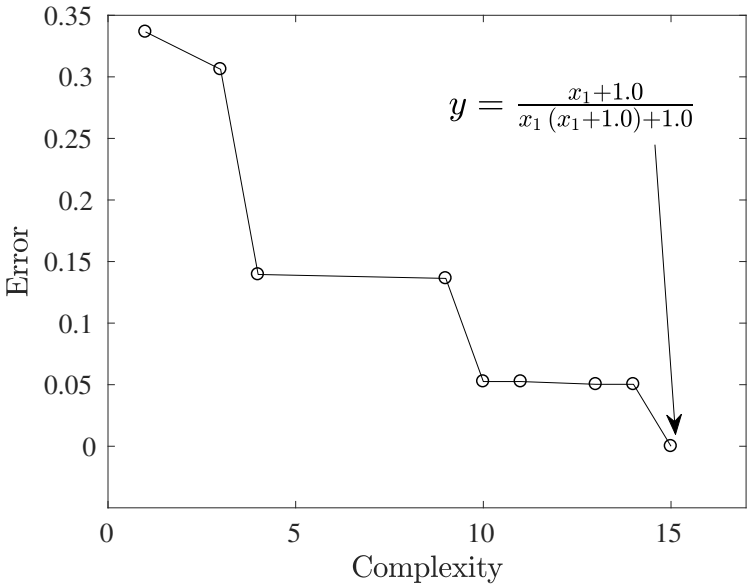


Figure 4.15: PO set of solutions found by CGP on solving the symbolic regression test problem of Section 4.5.2.

trary point along a *stream line* is:

$$y + \frac{v^2}{2g} + \frac{p}{\rho g} = c \tag{4.14}$$

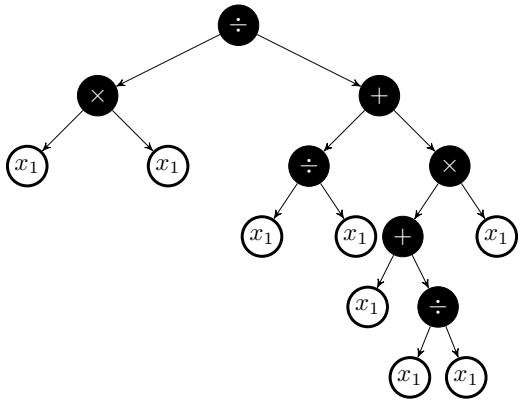


Figure 4.16: The tree corresponding to the chosen knee solution shown in Figure 4.15.

where

y is the elevation of the point above a reference plane

v is the fluid flow speed at a point on a streamline,

p is the pressure at the chosen point,

g is acceleration due to gravity,

ρ is the density of the fluid at all points in the fluid, and

c is a constant for the streamline chosen and also called *energy head*.

To generate data to test CGP , we assumed the fluid to be water ($\rho = 1000$ kg/m^3) and value of g to be 9.81 m/s^2 and we chose the value of $c = 20$ m of energy head for some streamline in some water flowing under steady flow conditions. We then randomly sampled 100 values each for the variables v and p from uniform distribution over [0,10] m/s and [101325,400000] Pa respectively. The atmospheric pressure at the surface of Earth is ≈ 101325 Pa. For these values of v, p, g, ρ and c, we then calculated 100 values of z using the relation

$$y = 20 - 0.051 \cdot v^2 - 1.0194 \times 10^{-4} \cdot p. \tag{4.15}$$

Note that Equation (4.15) is obtained by substituting the values of g, ρ and c assumed above in Equation (4.14). Considering y as the regressand variable and variables v and p as regressors, CGP was supplied with this data to learn the non-linear relation among these variables.

The CGP parameters used in solving this problem are given in Table 4.4. The PO solutions obtained by CGP for this problem are shown in Figure 4.17 in which each point represents a regressed non-linear relation between y and (v, p). Also shown in Figure 4.17 is the regressed rule for a knee solution. The trees corresponding to this term are shown

in Figure 4.18. The expression for the chosen solution is exactly same as the true relation shown in Equation (4.14).

Table 4.4: List of CGP parameters used for solving symbolic regression problem of Section 4.5.3.

Parameter	N	G	n_t	d_{max}	p_c	p_{ch}	p_{cl}	p_m
Value	28	30	6	4	0.95	0.20	0.80	0.05

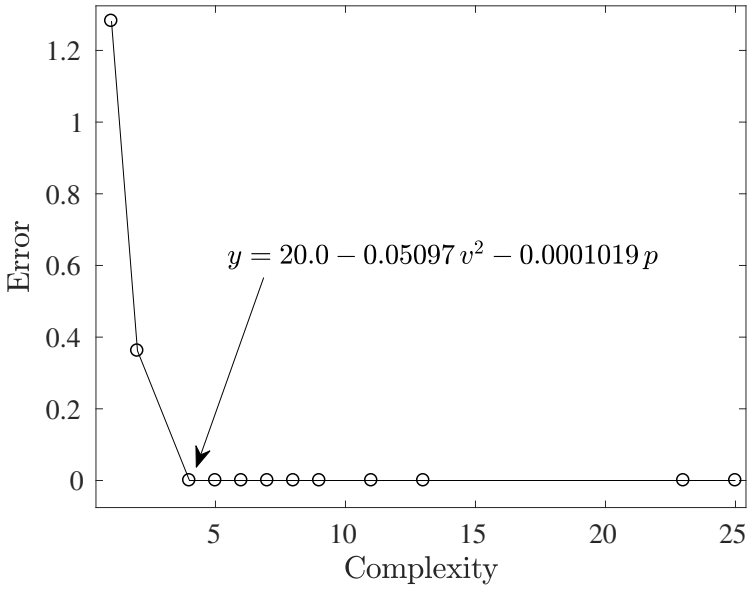
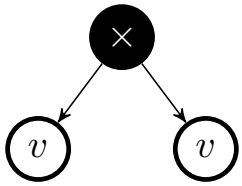


Figure 4.17: PO set of solutions found by CGP on solving the symbolic regression test problem of Section 4.5.3.

4.5.4 Test Problem-4

Let us look at the equation of deflection of a simply supported beam [100]. Consider a simply supported beam of length l having a mass per unit length of w. If E is the *Young's modulous* of the material of the beam and I is the area moment of inertia of the beam (about axis of bending), then the deflection of the beam at a distance x from one end is given by

$$\Delta_x = \frac{wx}{24EI}(l^3 - 2lx^2 + x^3). \tag{4.16}$$



(a) t_1

(b) t_2

Figure 4.18: The two trees corresponding to the chosen knee solution shown in Figure 4.17.

To generate data to test CGP , we assumed the beam to be made of steel with the following knowns parameters:

$$E = 29,000,000 \text{ psi},\ w = 0.568 \text{ lb/inch},\ I = 0.167 \text{ inch}^4,\ l = 39.37 \text{ inch}. \tag{4.17}$$

Substituting these values in Equation (4.16), we get

$$\Delta_x = 4.89 \times 10^{-9} x^4 - 3.85 \times 10^{-7} x^2 + 2.98 \times 10^{-4} x. \tag{4.18}$$

Using this relation, we then calculated 100 values of Δ_x based on 100 value of $x \in (0, l]$. Considering Δ_x as the regressand variable and variable x as the regressor, CGP was supplied with this data to learn the non-linear relation among these variables.

The CGP parameters used in solving this problem are given in Table 4.5. The PO solutions obtained by CGP for this problem are shown in Figure 4.19 in which each point represents a regressed non-linear relation between Δ_x and x. Also shown in Figure 4.19 is the regressed rule for a knee solution. The trees corresponding to this term are shown in Figure 4.20. The expression for the chosen solution is exactly same as the true relation shown in Equation (4.14).

Table 4.5: List of CGP parameters used for solving symbolic regression problem of Section 4.5.4.

Parameter	N	G	n_t	d_{max}	p_c	p_{ch}	p_{cl}	p_m
Value	30	30	6	6	0.95	0.20	0.80	0.05

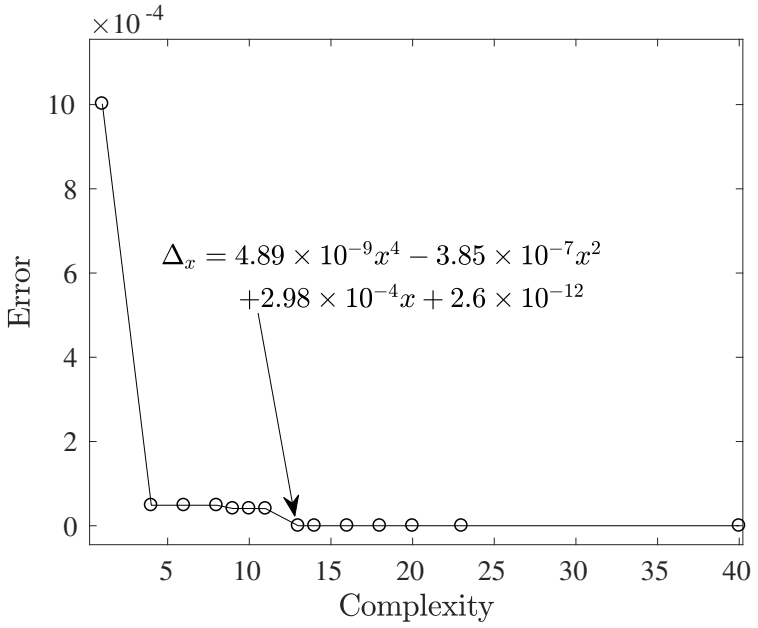


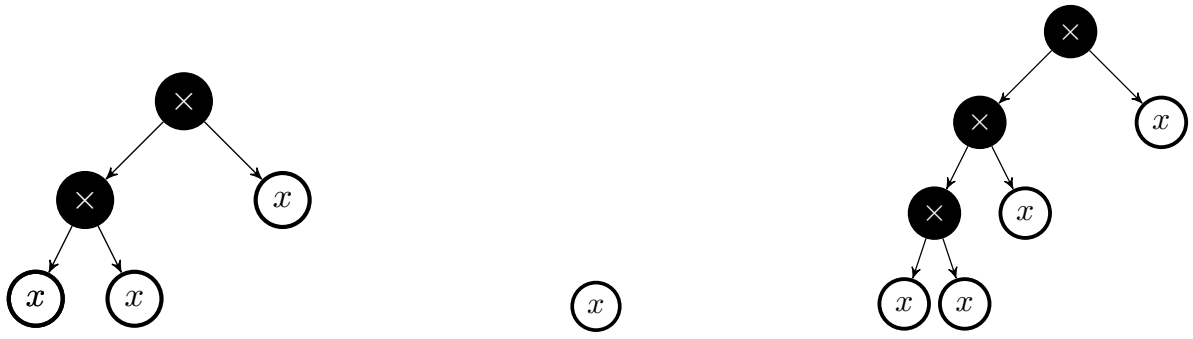
Figure 4.19: PO set of solutions found by CGP on solving the symbolic regression test problem of Section 4.5.4.

4.6 Noise Study

For the test problem of Section 4.5.3, a noise study was also performed. The purpose of this study was to understand, how much noise needs to be added to the regressand before the true relation is no longer present among the PO set of solutions of CGP . Gaussian white noise was added to the value of original regressand y as

$$\tilde{y} = y + a * N(0, 1), \tag{4.19}$$

where a or *noise factor* is some numeric value to change the level of noise and $N(0, 1)$ is the *standard normal* distribution. The noise levels are represented in terms of *signal to noise*



(a) t_1 (b) t_2 (c) t_3

Figure 4.20: The three trees corresponding to the chosen knee solution shown in Figure 4.19.

(SNR) ratio and measured in *decibels* or dBs using the formula

$$\text{SNR} = 10\log_{10}\left[\frac{\sum_{n=0}^{L-1} y^2(n)}{L \cdot a^2}\right] \text{ dB}, \tag{4.20}$$

where n is the index of observed value, L is the number of observations which is 100 in this case and a^2 is the variance of the signal $\tilde{y}$ having noise. This formula is taken from the work [101]. A high SNR means low noise. The value of regressors v and p were not changed. Effectively, CGP attempted to learn the relation

$$\tilde{y} = 20 - 0.051 \cdot v^2 + 1.0194 \times 10^{-4} \cdot p \tag{4.21}$$

instead of the one given by Equation (4.15). Note that the regressands are different in the two equations.

Table 4.6 shows the results of this study. The first column carries the value of noise factor a, the second column carries value of SNR in dBs, the third column carries the rule corresponding to the knee solution from among the PO set obtained by CGP on noisy data and the last column carries the R^2_{adj} value or *goodness of fit* value of the knee solution. This

study shows that the true solution is present among the PO set of solutions even until a high noise level of SNR = 15 dB, although the goodness of fit suffers a lot at that level of noise.

Table 4.6: Results of noise study performed on test problem of Section 4.5.3.

a	**SNR** (in dB)	**Knee soln rule**	R^2_{adj}
0.0010	100	$\tilde{y} = 20 - 0.051v^2 - 1.019 \times 10^{-4}p$	1
0.1925	30	$\tilde{y} = 20 - 0.051v^2 - 1.000 \times 10^{-4}p$	0.9882
0.3431	25	$\tilde{y} = 20 - 0.051v^2 - 9.9 \times 10^{-5}p$	0.9622
0.6090	20	$\tilde{y} = 19 - 0.051v^2 - 9.7 \times 10^{-5}p$	0.8484
1.0825	15	$\tilde{y} = 19 - 0.052v^2 - 9.2 \times 10^{-5}p$	0.6993
5.4315	1	$\tilde{y} = 28 - 6.3v + 1.9 \times 10^{-4}p + 2.6 \times 10^{-4}vp - 4.7v^2 + 0.72v^3 - 0.036v^4$	0.0125

4.7 Choosing a Solution

The results in the previous section show that CGP is competitive at symbolically regressing a set of PO rules that provide a good trade-off between prediction error and rule complexity. However, recall from Sections 2.5 and 2.5.3 that to perform online innovization, we not only have to learn the rules present in MOO data but also use that knowledge to modify solutions to expedite the EMO algorithm's convergence. Hence, if we use CGP for rule learning, once we have a set of PO solutions from CGP, we also need to choose a solution before passing the control back to the EMO algorithm. This can be done in two ways:

1. Choosing a *knee* solution : In PO solutions of bi-objective optimization problems, a knee point is a special solution. This is because choosing any solution other then the knee solution requires a large sacrifice in one objective to gain a small amount in the other objective. There is good amount of literature available on detecting and choosing a knee solution in a PO front [98, 102]. It can be seen from the PO fronts shown in Figures 4.12,4.15,4.17 and 4.19 that these problems show a strong propensity for knee

points. Hence choosing a knee solution in an automated way can be one way to choose one of the many PO solutions of CGP.

2. Checking *dimensional consistency* : Recall from Section 2.6 that one of the advantages of learning rules in form of mathematical algebraic expressions is that once can check if the learned rule adheres to the law of dimensional homogeneity. A rule that adds a physical quantity having the dimensions of length with a physical quantity having the dimensions of mass cannot add anything to the knowledge of a designer, even if rule has a low prediction error on training data. Hence, once CGP provides a set of PO solutions, we can check the dimensional consistency of those rules and discard the ones that violate it. We call this as *serial dimensional awareness* check because we are using the dimensionality check at the end of the CGP algorithm. Thus we can use this idea to shortlist some candidates from the PO set of CGP and then follow it up with a knee solution choosing method. Even in the absence of a knee in the PO front, it may be possible to parametrize the user preference in terms of rule complexity and prediction error and we can automatically choose a rule based on user's preference.

In the next chapter, we look at how can we check dimensional consistency of rules obtained by CGP.

Chapter 5

Using Dimensional Awareness with Rule Learning

For any physical law, adherence to the law of dimensional homogeneity is of utmost importance. The law of dimensional homogeneity [26] states that, "only commensurable quantities (physical quantities having the same dimension) may be compared, equated, added or subtracted". Although, the rule learning part of CGP can learn rules that accurately fit the data, but if any rule adds or subtracts two incommensurable quantities, then such a rule is physically meaningless. Hence, we need to quantify the degree of dimensional mismatch in a rule found by CGP. Such a quantification of dimensional mismatch for the PO rules found by rule learning part of CGP can give us additional information, if we need to choose only one or very few solutions out of the PO solutions of CGP.

5.1 Measuring Dimension Mismatch Penalty

Let us try to figure out how can we quantify dimensional mismatch penalty in a rule found by CGP. Say, the rule learning part of CGP is used for solving a symbolic regression problem relating regressand (y) and regressors ($x_k,\ k \in \{1, 2, \ldots, n_x\}$), which yields a set of PO rules.

Consider one such PO rule having the form

$$r \equiv y = w_0 + \sum_{i=1}^{n_t} w_i \cdot t_i, \tag{5.1}$$

where w_0 is a bias term, n_t is the total number of terms, w_i is the regression coefficient for term t_i and t_i is some function of regressors $x_k,\ k \in \{1, 2, \ldots, n_x\}$.

5.1.1 Case-I : Terms with Only Product and Division Operations

In this case, we will show how *dimensional mismatch penalty*, say $\mathcal{P}$, is calculated if the terms t_i in Equation (5.1) is comprised of only multiplication and division operations among the regressors $x_k,\ k \in \{1, 2, \ldots, n_x\}$. Let us further assume that;

- The number of fundamental dimensions present in data is three and they are the fundamental physical dimensions of mass (M), length (L) and time (T). Of course there can be more (for example temperature (θ), current (I etc.) but we are choosing aforementioned three for ease of representation.

- The derived physical dimensions of a term t_i is $\mathtt{M}^{\alpha_i}\mathtt{L}^{\beta_i}\mathtt{T}^{\gamma_i}$,

- $\mathcal{C}_j,\ j \in \{1, 2, \ldots, n_c\}$ are a set of n_c physical constants relevant to the process that is generating the data. These have to be chosen by subject matter experts. For example, when studying a fluid flow problem, some of the physical constants that may be considered important for the process are acceleration due to gravity, density of fluid and fluid viscosity. These $\mathcal{C}_j$ constants have dimensionless numeric values $c_j,\ j \in \{1, 2, \ldots, n_c\}$ and derived dimensions of $\mathtt{M}^{\lambda_j}\ \mathtt{L}^{\eta_j}\ \mathtt{T}^{\theta_j}$. The symbol $\mathcal{C}_j$ encapsulates both numeric value c_j as well as units information.

- To mend the dimensional inconsistencies between y and all terms t_i in Equation (5.1) using the constants $\mathcal{C}_j$'s, the constants may appear in dimensionally consistent form of Equation (5.1) with a limited set of exponents, say $\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$ where k is number of such chosen exponents. For example,

$$\mathcal{E} = \{-2.0,\ -1.0,\ -0.5,\ 0.0,\ 0.5,\ 1.0,\ 2.0\}. \tag{5.2}$$

- The fundamental units of the left hand side or regressand y in regressed rule shown in Equation (5.1) are $\mathtt{M}^{\varepsilon}\mathtt{L}^{\varphi}\mathtt{T}^{\omega}$.

If the derived physical dimensions of term t_i and regressand y are different, then a natural question to ask is, in what way can different physical constants $\mathcal{C}_j$ of the process be multiplied with the term t_i such that dimensional homogeneity can be maintained between t_i and y. One way to achieve this is as follows. For the i^{th} term t_i, there may exist a set of real values $\{z_{(i,1)},\ z_{(i,2)},\ \ldots,\ z_{(i,n_C)}\}$ such that a product of t_i with $\prod_{j=1}^{n_c} \mathcal{C}_j^{z_{(i,j)}}$ yields dimensional equivalence between term t_i and y. This can be represented as

$$\mathtt{M}^{\varepsilon}\mathtt{L}^{\varphi}\mathtt{T}^{\omega} = (\mathtt{M}^{\alpha_i}\mathtt{L}^{\beta_i}\mathtt{T}^{\gamma_i}) \prod_{j=1}^{n_C} (\mathtt{M}^{\lambda_j}\mathtt{L}^{\eta_j}\mathtt{T}^{\theta_j})^{z_{(i,j)}}, \tag{5.3}$$

which can be re-written as

$$\mathtt{M}^{(\varepsilon-\alpha_i)}\ \mathtt{L}^{(\varphi-\beta_i)}\ \mathtt{T}^{(\omega-\gamma_i)} = \mathtt{M}^{\sum_{j=1}^{n_C} \lambda_j z_{(i,j)}}\ .\ \mathtt{L}^{\sum_{j=1}^{n_C} \eta_j z_{(i,j)}}\ .\ \mathtt{T}^{\sum_{j=1}^{n_C} \theta_j z_{(i,j)}}. \tag{5.4}$$

Solving Equation (5.4) is equivalent to the system of simultaneous linear equations given by

$$\underbrace{\begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_{n_c} \\ \eta_1 & \eta_2 & \cdots & \eta_{n_c} \\ \theta_1 & \theta_2 & \cdots & \theta_{n_c} \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} z_{(i,1)} \\ z_{(i,2)} \\ \vdots \\ z_{(i,n_c)} \end{bmatrix}}_{\mathbf{z}_i} = \underbrace{\begin{bmatrix} \varepsilon - \alpha_i \\ \varphi - \beta_i \\ \omega - \gamma_i \end{bmatrix}}_{\mathbf{b}_i}, \tag{5.5}$$

From theory of linear algebra [103], solution $\mathbf{z}$ to Equation (5.5) is;

I The exact solution if $\mathbf{A}$ is full rank and square matrix,

II The least square solution if $\mathbf{A}$ is full rank and skinny matrix ($n_f > n_c$),

III The least square and least norm solution if $\mathbf{A}$ is full rank and fat matrix ($n_f < n_c$), and

IV The least square solution using *Singular Value Decomposition* method if $\mathbf{A}$ is singular.

Let the solution to Equation (5.5) be $\hat{\mathbf{z}}_i$ where

$$\hat{\mathbf{z}}_i = [\hat{z}_{(i,1)}\ \hat{z}_{(i,1)}\ \ldots\ \hat{z}_{(i,n_c)}]^\intercal \text{ where } \hat{z}_{(i,j)} \in \mathbb{R}\ \forall\ i, j. \tag{5.6}$$

Recall from Equation (5.3) that $z_{(i,j)}$s represent the exponents of the chosen physical constants $\mathcal{C}_j$s. Also, we are looking to *quantize* these exponents to a set of select few given by some set $\mathcal{E}$, an example of which is given by Equation (5.2). Hence, all the elements of $\hat{\mathbf{z}}_i$ are quantized to their nearest value in set $\mathcal{E}$. This quantization on the elements of $\hat{\mathbf{z}}_i$ yields

us the set $\bar{\mathbf{z}}_i$, where

$$\bar{\mathbf{z}}_i = [\bar{z}_{(i,1)}\ \bar{z}_{(i,2)}\ \ldots\ \bar{z}_{(i,n_c)}]^\intercal \text{ where } \bar{z}_{(i,j)} \in \mathbb{R}\ \forall\ i. \tag{5.7}$$

For example, if $\hat{\mathbf{z}}_i = \{-0.95\ \ 0.55\ \ 1.89\}$ and set $\mathcal{E}$ is given by Equation (5.2), then the quantized set will be given by

$$\bar{\mathbf{z}}_i = [-1.0\ 0.5\ 2.0]^\intercal.$$

Once a quantized set of exponents $\bar{\mathbf{z}}_i$ is obtained, we then obtain the residue vector $\mathbf{d}_i$ corresponding to the dimensional inconsistency in the i^{th} term in Equation (5.1) as

$$\mathbf{d}_i = \begin{bmatrix} d_{(i,1)} \\ d_{(i,2)} \\ d_{(i,3)} \end{bmatrix} = \begin{bmatrix} \varepsilon - \alpha_i \\ \varphi - \beta_i \\ \omega - \gamma_i \end{bmatrix} - \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_{n_c} \\ \eta_1 & \eta_2 & \cdots & \eta_{n_c} \\ \theta_1 & \theta_2 & \cdots & \theta_{n_c} \end{bmatrix} \begin{bmatrix} \bar{z}_{(i,1)} \\ \bar{z}_{(i,2)} \\ \vdots \\ \bar{z}_{(i,n_c)} \end{bmatrix}. \tag{5.8}$$

The *Root Mean Square* or RMS value of residue vector $\mathbf{d}_i$ can then be treated as penalty $\mathcal{P}_i$ of dimensional mismatch corresponding to the i^{th} term in Equation (5.1) as

$$\mathcal{P}_i = \sqrt{\frac{d_{(i,1)}^2 + d_{(i,2)}^2 + d_{(i,3)}^2}{3}}. \tag{5.9}$$

This dimensional mismatch penalty is a non-negative value and it will be zero if the physical dimensions of y and term t_i in Equation (5.1) can be matched exactly by multiplying t_i with just the right combination of physical constants, $\mathcal{C}_j$ where $j \in \{1, 2, \ldots, n_c\}$, when raised to a particular set of exponents given by $\bar{\mathbf{z}}_i$ in Equation (5.7). Once this penalty value can be calculated for all n_t the terms of Equation (5.1), we can calculate the overall dimensional

mismatch penalty $\mathcal{P}$ for the expression given by Equation (5.1) as

$$\mathcal{P} = \begin{cases} \dfrac{\sum_{i=1}^{n_t} \mathcal{P}_i}{n_t - \sum_{i=1}^{n_t} \delta_{\mathcal{P}_i,0}} & \text{if } \sum_{i=1}^{n_t} \delta_{\mathcal{P}_i,0} > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{5.10}$$

where $\delta_{i,j}$ is the Kronecker delta function such that

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{5.11}$$

If Equation (5.10) evaluates to zero, then it implies that the original equation can be made dimensionally consistent using the set of chosen constants $\mathcal{C}_j$s and a set of exponents such as given by (5.7). In such a case, its beneficial to modify the original equation given by (5.1) to include the chosen constants as

$$r \equiv y = w_0 + \sum_{i=1}^{n_t} \left(\hat{w}_i \cdot t_i \prod_{j=1}^{n_c} \mathcal{C}_j^{\bar{z}(i,j)} \right), \text{ where} \tag{5.12}$$

$$\hat{w}_i = \frac{w_i}{\prod_{j=1}^{n_c} c_j^{\bar{z}(i,j)}}. \tag{5.13}$$

This is done so that we do not affect the regression or classification accuracy of the original equation while making it dimensionally consistent at the same time. Recall that c_j represents the dimensionless numerical value of some physical constant $\mathcal{C}_j$ and the symbol $\mathcal{C}_j$ encapsulates both numeric value c_j as well as units information.

5.1.2 Dimensionally Inconsistent Example

Let us understand the above procedure with an example. Consider the symbolic regression problem (Bernoulli equation case) presented in Section 4.5.3. We chose variable y as the regressand and variable v and p as regressors. The rule learning part of CGP returns a set of PO rules as shown in Figure 4.17. Suppose we want to check the dimensional consistency of a one of the PO solutions given by

$$r \equiv y = \underbrace{20}_{w_0} - \underbrace{0.05097}_{w_1} \cdot \underbrace{v^2}_{t_1} - \underbrace{0.0001019}_{w_2} \cdot \underbrace{p}_{t_2} + \underbrace{3.9 \times 10^{-8}}_{w_3} \cdot \underbrace{v}_{t_3}. \tag{5.14}$$

The rule shown in Equation (5.14) is different from the knee solution shown in Figure 4.17. In this rule, there are three terms ($n_t = 3$) namely, $t_1 = v^2$, $t_2 = p$ and $t_3 = v$ and their corresponding weights are $w_1 = -0.050907$, $w_2 = -0.0001019$ and $w_3 = 3.9 \times 10^{-8}$ respectively. Also, there is a bias term $w_0 = 20$. Recall that the units of v and p are m/s and Pa respectively. Therefore, the derived units for t_1 are $\mathtt{M}^0\mathtt{L}^2\mathtt{T}^{-2}$, for t_2 are $\mathtt{M}^1\mathtt{L}^{-1}\mathtt{T}^{-2}$ and for t_3 are $\mathtt{M}^0\mathtt{L}^1\mathtt{T}^{-1}$.

Let us choose two physical constants ($n_c = 2$) in the Bernoulli problem, namely acceleration due to gravity, 'g' and density of fluid (water here), ρ. These are measured in m/s^2 and kg/m^3 respectively. Hence,

$$g \equiv \mathcal{C}_1 = \underbrace{9.81}_{c_1}\ m/s^2 \text{ and },\tag{5.15}$$

$$\rho \equiv \mathcal{C}_2 = \underbrace{1000}_{c_2}\ kg/m^3. \tag{5.16}$$

The derived units for these physical constants of the problem are $\mathtt{M}^0\mathtt{L}^1\mathtt{T}^{-2}$ for g and $\mathtt{M}^1\mathtt{L}^{-3}\mathtt{T}^0$

for ρ. Furthermore, let us choose the set given in (5.2) as the set of allowed exponents for the constants.

Since y is the elevation of a point on a streamline in fluid, it is measured in meters. Thus, the derived dimensions of regressand y is given by $\mathtt{M}^0\mathtt{L}^1\mathtt{T}^0$. This information is summarized below.

$$\begin{array}{lll} \alpha_1 = 0 & \beta_1 = 2 & \gamma_1 = -2 \\ \alpha_2 = 1 & \beta_2 = -1 & \gamma_2 = -2 \\ \alpha_3 = 0 & \beta_3 = 1 & \gamma_3 = -1 \\ \lambda_1 = 0 & \eta_1 = 1 & \theta_1 = -2 \\ \lambda_2 = 1 & \eta_2 = -3 & \theta_2 = 0 \\ \varepsilon = 0 & \varphi = 1 & \omega = 0 \end{array}$$

Let us look at the dimensional consistency of each term one by one.

5.1.2.1 First Term

For term t_1 in Equation (5.14), Equation (5.5) is given by

$$\underbrace{\begin{bmatrix} 0 & 1 \\ 1 & -3 \\ -2 & 0 \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} z_{(1,1)} \\ z_{(1,2)} \end{bmatrix}}_{\mathbf{z}_1} = \underbrace{\begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix}}_{\mathbf{b}_1},$$

which upon solving and quantizing to allowed exponent values yields

$$\bar{\mathbf{z}}_1 = [-1\ 0]^\intercal.$$

Substituting $\bar{z}_1$ in Equation (5.8) and then using Equation (5.9), we obtain the dimensional mismatch penalty for first term of Equation (5.14) as

$$\mathcal{P}_1 = 0. \tag{5.17}$$

5.1.2.2 Second Term

For term t_2 in Equation (5.14), Equation (5.5) is given by

$$\underbrace{\begin{bmatrix} 0 & 1 \\ 1 & -3 \\ -2 & 0 \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} z_{(2,1)} \\ z_{(2,2)} \end{bmatrix}}_{\mathbf{z}_2} = \underbrace{\begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}}_{\mathbf{b}_2},$$

which upon solving and quantizing to allowed exponent values yields

$$\bar{\mathbf{z}}_2 = [-1\ -1]^\intercal.$$

Substituting $\bar{z}_2$ in Equation (5.8) and then using Equation (5.9), we obtain the dimensional mismatch penalty for second term of Equation (5.14) as

$$\mathcal{P}_2 = 0. \tag{5.18}$$

5.1.2.3 Third Term

For term t_3 in Equation (5.14), Equation (5.5) is given by

$$\underbrace{\begin{bmatrix} 0 & 1 \\ 1 & -3 \\ -2 & 0 \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} z_{(3,1)} \\ z_{(3,2)} \end{bmatrix}}_{\mathbf{z}_3} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{\mathbf{b}_3},$$

which upon solving yields

$$\hat{\mathbf{z}}_3 = [-0.4878 \ -0.1463]^\intercal. \tag{5.19}$$

Since we chose the allowed set of exponents to be

$$\mathcal{E} = \{-2.0, \ -1.0, \ -0.5, \ 0.0, \ 0.5, \ 1.0, \ 2.0\},$$

hence quantizing the exponents in (5.19) to values in $\mathcal{E}$ yields

$$\bar{\mathbf{z}}_3 = [-0.5 \ 0.0]^\intercal.$$

Substituting $\bar{z}_3$ in Equation (5.8) and then using Equation (5.9), we obtain the dimensional mismatch penalty for third term of Equation (5.14) as

$$\mathcal{P}_3 = 0.2887. \tag{5.20}$$

The total dimensional penalty for symbolic regression solution given by (5.14) is found using Equation (5.10), i.e.

$$\mathcal{P} = \frac{\sum_{i=1}^{n_t} \mathcal{P}_i}{n_t - \sum_{i=1}^{n_t} \delta_{\mathcal{P}_i,0}} = \frac{0+0+0.2887}{3-(1+1+0)} = 0.2887 \neq 0 \tag{5.21}$$

This should be the case because if we compare Equation (5.14) with the correct Bernoulli equation of Equation (4.15), the third term of Equation (5.14) is not commensurate with the physical dimensions of the regressand y.

5.1.3 Dimensionally Consistent Example

Again consider the symbolic regression problem presented in Section 4.5.3. Suppose that the rule learning part of CGP returns a set of PO rules, and we want to check the dimensional consistency of one such rule given by

$$r \equiv y = \underbrace{20}_{w_0} - \underbrace{0.05097}_{w_1} \cdot \underbrace{v^2}_{t_1} - \underbrace{0.0001019}_{w_2} \cdot \underbrace{p}_{t_2}. \tag{5.22}$$

Equation (5.22) is actually a knee region solution among the PO solutions found by CGP in Bernoulli test data, shown in Figure 4.17. From Sections 5.1.2.1 and 5.1.2.2, we know that this equation can be made dimensionally consistent using the chosen physical constants. The first chosen constant (acceleration due to gravity) $\mathcal{C}_1$ is symbolically represented by g and numerically equal to $c_1 = 9.81\ m/s^2$ in SI units. The second chosen constant (density of water) $\mathcal{C}_2$ is symbolically represented by ρ and numerically equal to $c_2 = 1000\ kg/m^3$ in SI units. Hence, we can re-evaluate the weights in Equation (5.22) to include these chosen phys-

ical constants using Equations (5.12) and (5.13). Substituting values from Equations (5.17), (5.18) and (5.22) in Equation (5.13), we get

$$\hat{w}_1 = \frac{w_1}{c_1^{\bar{z}_{(1,1)}} c_2^{\bar{z}_{(1,2)}}} = \frac{-0.05097}{9.81^{-1}\ 1000^{0}} = -0.50001 \text{ and} \tag{5.23}$$

$$\hat{w}_2 = \frac{w_2}{c_1^{\bar{z}_{(2,1)}} c_2^{\bar{z}_{(2,2)}}} = \frac{-0.0001019}{9.81^{-1}\ 1000^{-1}} = -0.99964. \tag{5.24}$$

Using Equation (5.12), the modified form of Equation (5.22) that includes the chosen physical constants can be written as

$$\begin{aligned} r \equiv y &= 20 + \hat{w}_1 \cdot v^2\ \mathcal{C}_1^{\bar{z}_{(1,1)}}\ \mathcal{C}_2^{\bar{z}_{(1,2)}} + \hat{w}_2 \cdot p\ \mathcal{C}_1^{\bar{z}_{(2,1)}}\ \mathcal{C}_2^{\bar{z}_{(2,2)}}, \text{ or} \\ &= 20 + \hat{w}_1 \cdot v^2\ g^{-1}\ \rho^{0} + \hat{w}_2 \cdot p\ g^{-1}\ \rho^{-1} \\ &= 20 - 0.50001\frac{v^2}{g} - 0.99964\frac{p}{\rho g} \text{ (upon substituting } \hat{w}_1 \text{ \& } \hat{w}_2). \end{aligned} \tag{5.25}$$

Compare Equation (5.25) with known form of Bernoulli's equation given in Equation (4.14) and notice that two are same (within a small tolerance) if $c = 20$. Figure 5.1 shows the PO solutions of the Bernoulli equation problem shown in Figure 4.17 along with the dimension mismatch penalty information. The solutions having a non-zero penalty value are shown in red. Note that only the knee solution has a zero dimension mismatch penalty in this case. Hence, we can use dimensional mismatch penalty at the end of a CGP run to discard PO solutions which do not adhere to law of dimensional homogeneity.

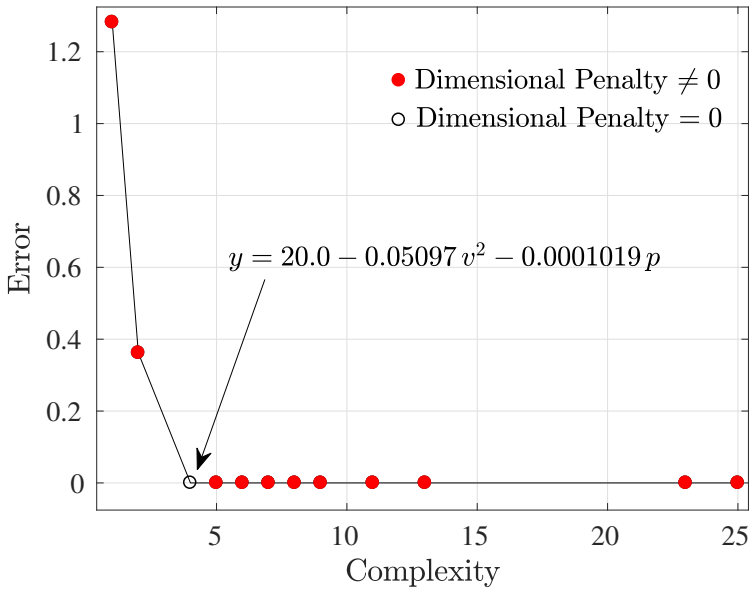


Figure 5.1: PO set of solutions found by CGP on solving the symbolic regression test problem of Section 4.5.3 followed by a dimensional penalty calculation for each case. Only the knee solution, which is also the exact solution, has a dimensional mismatch penalty of zero.

5.1.4 Case-II : More Complex Terms

In Section 5.1.1, we limited our study to cases where the term t_i in Equation (5.1) is only composed of product and division operators among the regressors. However as shown in Sections 4.5.1, 4.5.2 and 4.5.3, CGP is capable of much more, including rational functions and functions involving addition and subtraction operators. In such a case, we have to make a slight change to the procedure outlined in Section 5.1.1 for calculating dimensional penalty values for a term t_i of Equation (5.1). We know that in Equation (5.1),

$$t_i = f(x_1, x_2, \ldots, x_{n_x}) \tag{5.26}$$

where $x_i \in \{1, 2, \ldots, n_x\}$ are the regressors and function $f()$ is some function involving the operations of addition, subtraction, division and multiplication. Upon substituting the physical dimensions of the regressors in (5.26), lets say that the derived physical dimensions

of term t_i are obtained as some rational function of the fundamental dimensions as

$$\frac{P(\mathtt{M},\mathtt{L},\mathtt{T})}{Q(\mathtt{M},\mathtt{L},\mathtt{T})} \tag{5.27}$$

where $\mathcal{P}()$ and and $\mathcal{Q}()$ are some polynomials in M,L and T. Using partial fractions, we can decompose (5.27) as

$$\frac{P(\mathtt{M},\mathtt{L},\mathtt{T})}{Q(\mathtt{M},\mathtt{L},\mathtt{T})} = \sum_i \frac{p_i(\mathtt{M},\mathtt{L},\mathtt{T})}{q_i(\mathtt{M},\mathtt{L},\mathtt{T})}, \tag{5.28}$$

such that

- The denominator, $q_i(\mathtt{M},\mathtt{L},\mathtt{T})$, of each fraction is a power of an irreducible (not factorable into polynomials of positive degree) polynomial and
- The numerator, $p_i(\mathtt{M},\mathtt{L},\mathtt{T})$, is a polynomial of smaller degree than that of $q_i(\mathtt{M},\mathtt{L},\mathtt{T})$.

In such a case, both $p_i()$ and $q_i()$ are polynomials in M,L and T that cannot be broken down further. Let us fully expand both of these polynomials as sums of product terms, then we can re-write Equation (5.28) as

$$\sum_i \frac{p_i(\mathtt{M},\mathtt{L},\mathtt{T})}{q_i(\mathtt{M},\mathtt{L},\mathtt{T})} = \sum_i \frac{\sum\limits_j r_{i,j}(\mathtt{M},\mathtt{L},\mathtt{T})}{\sum\limits_k s_{i,k}(\mathtt{M},\mathtt{L},\mathtt{T})}. \tag{5.29}$$

In Equation (5.29), each term $r_{i,j}(\mathtt{M},\mathtt{L},\mathtt{T})$ and $s_{i,k}(\mathtt{M},\mathtt{L},\mathtt{T})$ are purely product terms in M,L,T. In the CGP code, we implement this using the *Symbolic Math Toolbox* of MATLAB.

Now we can use the method of Section 5.1.1 to find the dimension mismatch penalty value for each product term separately. The only difference being that now the dimensions of the numerator terms, $r_{i,j}(\mathtt{M},\mathtt{L},\mathtt{T})$, have to be matched with those of numerator of the left hand side of Equation (5.1) (i.e. y). Similarly, the dimensions of the denominator

terms, $s_{i,k}(\mathtt{M}, \mathtt{L}, \mathtt{T})$, have to be matched with those of denominator of the left hand side of Equation (5.1) (i.e. 1). Summing together the penalty values for all the numerator and denominator terms gives us the total dimension mismatch penalty associated with a term t_i that has addition and subtraction operators as well combined with product and division operators.

Until now, we have looked at how can we use CGP for a symbolic regression task and then follow it up with a dimension mismatch check on the PO solutions. In the next chapter, we will look at using CGP as a binary classifier and produce the decision boundaries as algebraic expressions. Note that the procedure explained in this chapter will be applicable even when CGP is being used for a classification task. The only difference in that case will be that the regressand y will be replaced by a class label $\mathcal{Y}$ which is dimensionless. Lets now move on to another interesting capability of CGP, i.e. solving binary classification tasks.

Chapter 6

Learning Free Form Rules Using a Classification Task

In this chapter, we will learn how to use the CGP developed in Chapter 4 for a classification task. Recall from Sections 2.4 and 2.5 that when solving a MOO using an EMO algorithm, knowledge can be derived in one of two ways, namely:

- By searching for rules in some preferred set of solutions, say the non-dominated set. The methods of rule learning developed in Chapters 3 and 4 can be used in this case.
- By searching for discriminatory rules between a preferred set and an unpreferred set of solutions, say non-dominated versus dominated solutions set. This is the case of rule learning that we will address in this chapter.

We will present as to how we can use the CGP developed in Chapter 4 for a classification task. This work was developed as part of an industry project. The methodology developed as part of this project is directly applicable to task of online innovization as well for learning decision boundary in a classification task as an algebraic expression in terms of features. Let us quickly learn about the classification problem first before learning about how to use CGP for a classification task.

6.1 Target Classification Problem

This goal of this project was to develop a computationally efficient machine learning methodology that can:

- Automate the process of selecting a few important features from a set of features and then building a classifier using those features,
- And learn the classifier (decision boundary) as an algebraic expression involving the selected important features.

The data is being produced by a fast manufacturing process in which it is being captured as a multi-variate time series data via many sensors. This time series data is then processed to extract many features using basic mathematical functions such as differentiation and integration without any expert knowledge for feature creation. This part of feature extraction is not being shared in this work because of non-disclosure agreement with the industry partner.On similar grounds, we will not describe the exact manufacturing process as well. However, our method of extracting rules based classifier is applicable to any manufacturing process where we are collecting a lot of data about the process while the process is still going on. For example, if we are collecting multiple sensors data for a welding process such as welding current, voltage and distance of electrode from the weld region.

Reiterating, that the term 'interpretable-rules' in the context of this research refers to rules in the form of mathematical expressions/equations involving the process features, process constants and some simple operations such as addition, subtraction, multiplication and division. The term 'meaningful-rules' in the context of this research refers to the idea of aforementioned expressions being physically meaningful by being dimensionally consistent. Now, let us now look as how can we use the CGP developed in Chapter 4 for a classification

task.

6.2 Using CGP for a Classification Task

In Chapter 4, we saw how CGP learns a rule of the form given by Equation (4.1), i.e.

$$\psi(\phi_1, \phi_2, \ldots, \phi_{n_\phi}) \equiv \phi_i = w_0 + \sum_{j=1, j\neq i}^{n_t} w_j \cdot t_j,$$

where ψ represents one such rule, ϕ are the basis functions explained in Section 3.1 that represent the data from MOO problem, ϕ_i is one of the basis functions that is a regressand, rest of the basis functions are regressors and we use CGP as a symbolic regression tool to learn the aforementioned form of rules. In this case, CGP optimizes the structure of rules and learns the weights using OLSR. However, if the basis function ϕ_i is a class label instead of a regressand, then we can use a linear *support vector machine* (SVM) [56] learning algorithm for learning the weights. This is because the results of linear SVM are considered very interpretable. The challenge lies in finding the right number of higher dimensions (of feature space) and the right features/*derived-features* corresponding to those dimensions, in which the data is linearly separable. In such a space, a linear SVM will be able to find out an appropriate separation plane with relative ease, provided the that the decision boundary is not discontinuous. By derived features, we mean features that are composed from the initial set of features provided to CGP using basic operations such as addition, subtraction, multiplication and division. Let us look at an example.

Consider the binary data shown in Figure 6.1 which is generated using the following

equation of an ellipse

$$y = -x_1^2 + 2.02x_1 \cdot x_2 - 3.05x_2^2 + 1.98 = 0, \tag{6.1}$$

where x_1 and x_2 are the two features for this data. The data of hypothetical *Go* class ($y < 0$) is shown in green and the data of hypothetical *NoGo* class ($y \geq 0$) is shown in red. Clearly, Equation (6.1) defines the decision boundary for this problem. Upon trying to classify this

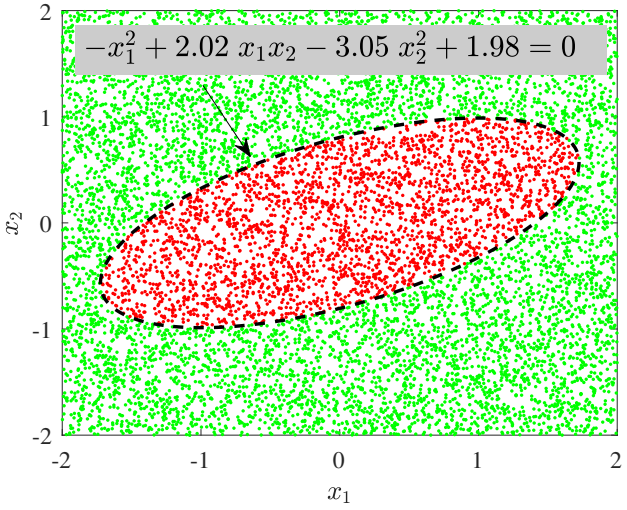


Figure 6.1: A hand crafted example of binary class data.

data set using a tree classifier, the learned tree model has 13 levels and 147 nodes as shown in Figure 6.2. What is interesting to note is that, if we provide only the features x_1 and x_2 to a linear SVM algorithm, it will perform very poorly as the data is not *linearly separable*. Now consider the following three features, namely x_1^2, x_2^2 and $x_1 \cdot x_2$. We call these three features as *derived features* as they were not provided with the original features of the problem but are derived from the same. Now if we provide these three features to a linear SVM algorithm, it will perform exceedingly well on the same data. The reason being that in this modified 3-Dimensional feature space, the data is linearly separable. This can be seen in Figure 6.3

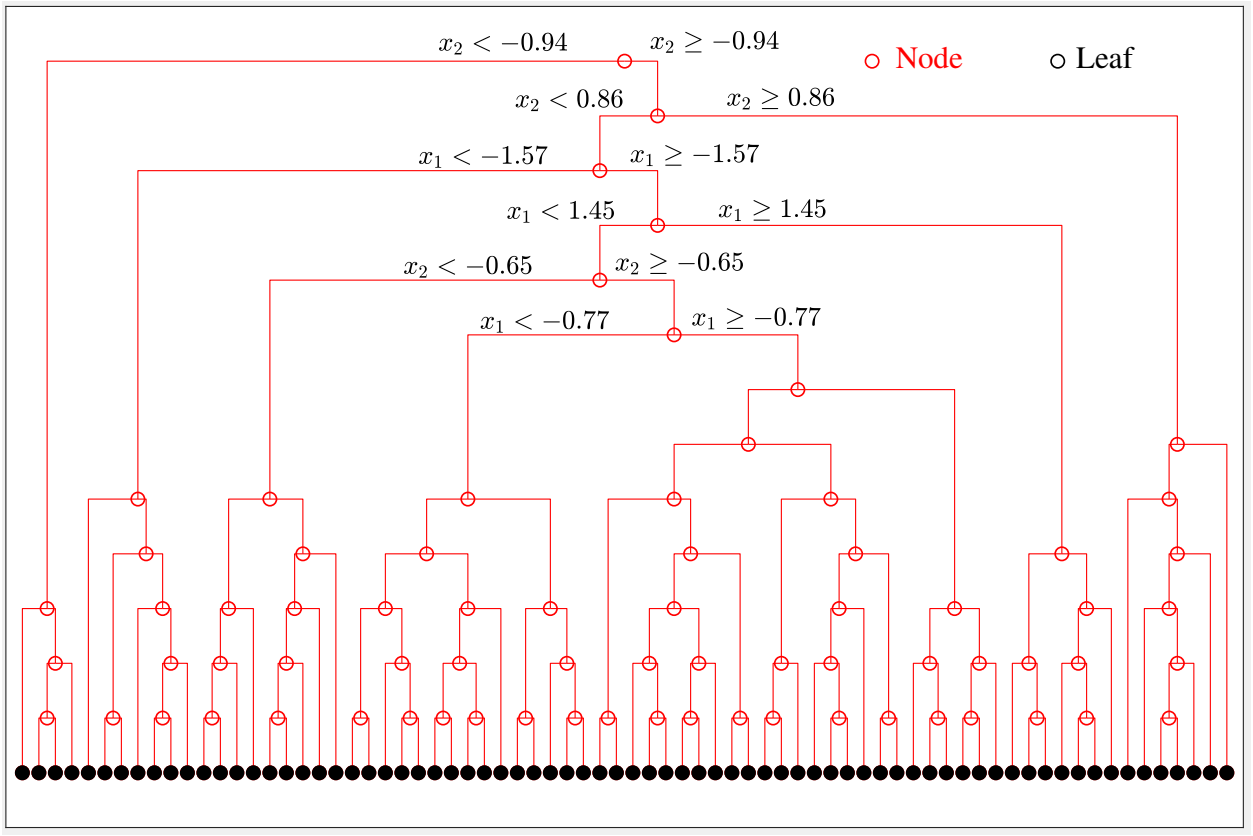
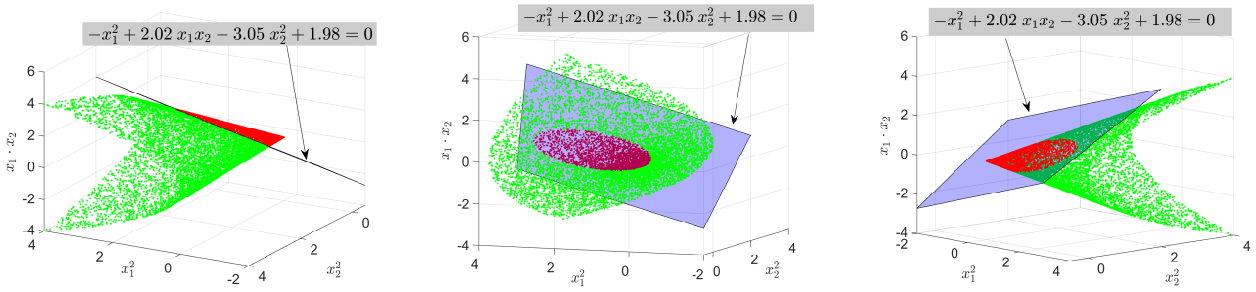


Figure 6.2: Binary classification tree model for classifying the two class data shown in Figure 6.1.

where the sub-figures show the same data in the derived feature space from three different camera angles.

As can be seen in Figures 6.3a, 6.3b and 6.3c, a linear SVM could find a plane (in blue) clearly separating the *Go* and *NoGo* data. Not surprisingly, the equation of the plane is same as Equation (6.1).

One may argue, why don't we use a quadratic or polynomial kernel based SVM for this problem. That may work in this problem but what if the problem requires a rational function as one of the dimensions of the expanded feature space? Furthermore, the choice of what kernel to select with keeping the interpretability of the classifier/decision-boundary in mind is not very straight forward. Also, having the decision boundary in terms of some algebraic



(a) Camera angle-1 (b) Camera angle-2 (c) Camera angle-3

Figure 6.3: The binary class data of Figure 6.1 shown in a derived feature space where the data is linearly separable.

equation in terms of features and not some kernel which corresponds to an infinite dimensional feature space such as *radial basis functions* helps us later in checking the dimensional consistency of learned model. Maintaining dimensional consistency is an important handle available with engineers to do a sanity check of models learned for some physical process or system.

6.2.1 Evaluating Fitness of a CGP Individual for a Binary Classification Task

Consider a classification problem with n_o observations, n_x number of features x_i, and n_o binary class labels ($y_j \in \{0, 1\}$) initially provided with the problem. When solving a classification problem using CGP, consider a CGP individual with same rule structure as shown in Equation(5.1), i.e.

$$r \equiv y = w_0 + \sum_{i=1}^{n_t} w_i \cdot t_i,$$

where n_t is the number of terms in the rule. The terms t_i can be considered as derived features obtained by simple operations of $\{+, -, \times, \div\}$ on the original features. The weights of this individual are then learned using a linear SVM method and the miss-classification

error rate at the end of weight optimization by SVM is assign as error fitness to the individual. This is measured as

$$f_{\text{error}} = e_I^{(trn)} + e_{II}^{(trn)} \tag{6.2}$$

where $e_I^{(trn)}$ and $e_{II}^{(trn)}$ represent the type-I and type-II error rates (in %) [104] on the training set. The complexity fitness is calculated same as in case of the symbolic regression case given by Equation (4.11), i.e.

$$f_{\text{complexity}} = \sum(\text{ Nodes in all terms of CGP individual}).$$

Furthermore, in case of industry's classification problem, the cost of miss-classifying *NoGo* product was much more then the cost of miss-classifying a *Go* product. For this reason, the cost matrix used by the linear SVM for arriving at the weights is kept to be:

$$C = \begin{bmatrix} 0 & 1 \\ 25 & 0 \end{bmatrix},$$

i.e. cost of making type-II error on the training set is set 25 times higher than cost of making a type-I error.

6.3 Performance on Small Feature Space

Let us now look at some results obtained for classifying real production data. We chose production data from two dates for our study. We will name these two data sets as data set-1 and data set-2. The details of the production data from these two days is given in Table 6.1. A total of ten features, namely x_i such that $i \in \{1, \ldots, 10\}$ were extracted.

Table 6.1: Production data details used for testing CGP classifier.

Date	**# of *Go* products**	**# of *NoGo* products**
Data set-1	2381	6
Data set-2	1882	6

Table 6.2 shows the physical dimensions of these ten features. Due to highly imbalanced datasets, we used *adaptive smote* method [105, 106] to oversample the minority *NoGo* class.

Table 6.2: Small feature set and their details.

Feature(s)	Physical Dimension
x_1, x_2, x_3	$\mathrm{L}^2\ \mathrm{M}^1\ \mathrm{T}^{-2}$
x_4	$\mathrm{L}^1\ \mathrm{M}^0\ \mathrm{T}^0$
x_5, x_6	$\mathrm{L}^1\ \mathrm{M}^0\ \mathrm{T}^{-1}$
x_7	$\mathrm{L}^0\ \mathrm{M}^0\ \mathrm{T}^0$
x_8, x_9, x_{10}	$\mathrm{L}^0\ \mathrm{M}^0\ \mathrm{T}^{-1}$

6.3.1 Results on Production Data Set-1

Here we discuss the CGP results for production data set-1. Training was conducted over 70% of data. Table 6.3 shows the CGP parameters used in this case. Figure 6.4 shows the

Table 6.3: List of CGP parameters used for solving binary classification problem of Section 6.3.1.

Parameter	N	G	n_t	d_{max}	p_c	p_{ch}	p_{cl}	p_m
Value	200	200	10	6	0.85	0.20	0.80	0.15

set of PO classifiers obtained for production data of data set-1. The vertical axis of the figure represents *misclassification error* in percent of training data set and the horizontal axis represents the complexity of a decision rule.

Three solutions are highlighted with different colors with some extra information about the corresponding classifier. These three solutions/classifiers represent three different trade-

offs with respect to accuracy and complexity, starting with a classifier which is simplest but most inaccurate (shown in blue), to a solution with intermediate values of classification error and complexity (shown in red), and finally a solution which is very complex but highly accurate (shown in brown). For each of these solutions, we have also shown the type-I and type-II error obtained on the test data set. Table 6.4 shows this information in a tabular form. Note that for all three solutions shown in Table 6.4, only six features of the original ten features appear in the discovered classification rules. The features x_7, x_8, x_9 and x_{10} do not appear in any of these solutions.

Furthermore, Figure 6.5 shows the decision boundary and segregation of the *Go* and *NoGo* classes in the feature space for the knee solution classifier of Figure 6.4. The *NoGo* class shown in Figure 6.5 includes both real and synthetic *NoGo* class data of data set-1. Interestingly, the CGP algorithm is able to find the right feature space, namely $t_1 = \frac{x_3}{x_6}$, $t_2 = \frac{x_5}{x_4}$ and $t_3 = \frac{x_1}{x_3 \cdot x_4}$, in which the production data is linearly separable.

Table 6.4: Summary of classification rules found for binary classification problem of Section 6.3.1 and their corresponding error rates on training and test sets.

Soln Desc.	**Rule for *NoGo* products**	e_I^{trn}	e_{II}^{trn}	e_I^{test}	e_{II}^{test}
Simplest	$-20.08 + 7.385\,\frac{x_2}{x_4} > 0$	30.90%	0.00%	32.18%	0.00%
Knee	$-25.72 + \frac{1.348\,x_3}{x_6} - \frac{3.028\,x_5}{x_4} + \frac{3.141\,x_1}{x_3\,x_4} > 0$	3.61%	0.00%	4.57%	0.00%
Most Accurate	$-34.76 - \frac{0.23\,x_5}{x_1 x_4^2} - 0.53x_3 + \frac{0.65x_2}{x_4x_6} + \frac{0.17x_5}{x_4x_6} + \frac{4.07x_1}{x_4(x_3 - x_4)} > 0$	0.36%	0.00%	1.25%	0.0%

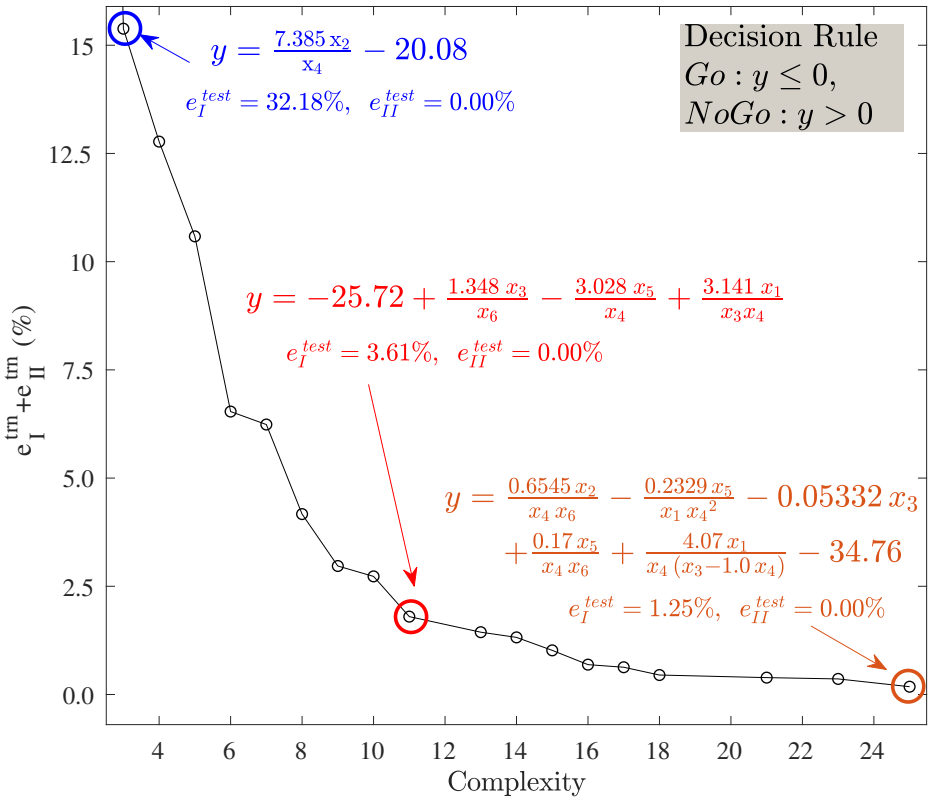


Figure 6.4: PO set of solutions found by CGP on solving the binary classification problem of Section 6.3.1.

6.3.2 Results on Production Data Set-2

Here we discuss the CGP results for production data from data set-2. Training was conducted over 70% of data and rest of the data was kept unseen to the training stage of CGP. Table 6.5 shows the CGP parameters used in this case. Figure 6.6 shows the set of PO classifiers obtained for production data of data set-2. The vertical axis of the figure represents *misclassification error* in percent of training data set and the horizontal axis represents the complexity of a decision rule. Notice that CGP has performed relatively better in terms of

Table 6.5: List of CGP parameters used for solving binary classification problem of Section 6.3.2.

Parameter	N	G	n_t	d_{max}	p_c	p_{ch}	p_{cl}	p_m
Value	100	100	10	6	0.85	0.20	0.80	0.15

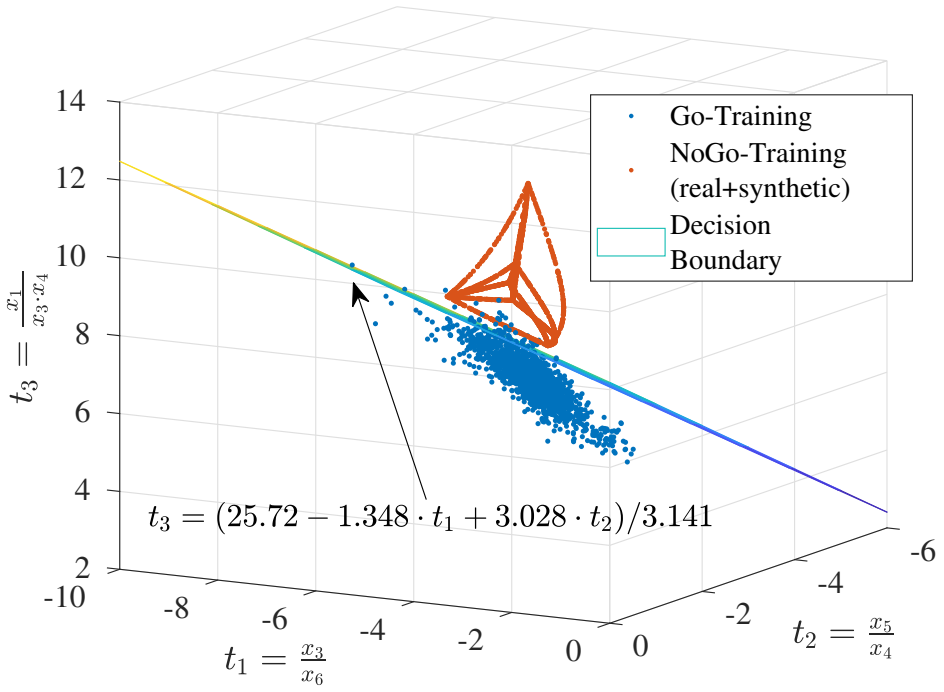


Figure 6.5: Decision boundary for the knee solution classifier of PO set of classifiers shown in Figure 6.4.

finding very accurate classifiers on data from data set-2 as compared to classifiers learned on data from data set-1.

Again, we have highlighted three solutions with different colors and we provide some extra information about the corresponding classifier. These three solutions/classifiers represent three different trade-offs with respect to accuracy and complexity, starting with a classifier which is simplest but most inaccurate (shown in blue), to a solution with intermediate values of classification error and complexity (shown in red), and finally a solution which is very complex but highly accurate (shown in brown). For each of these solutions, we have also shown the type-I and type-II error obtained on the test data set. Table 6.6 shows the same information in a tabular form.

One result that may bother a careful eye is that of the error rates obtained for the knee solutions classifier shown in Table 6.6. As shown, both type-I and type-II errors on the test

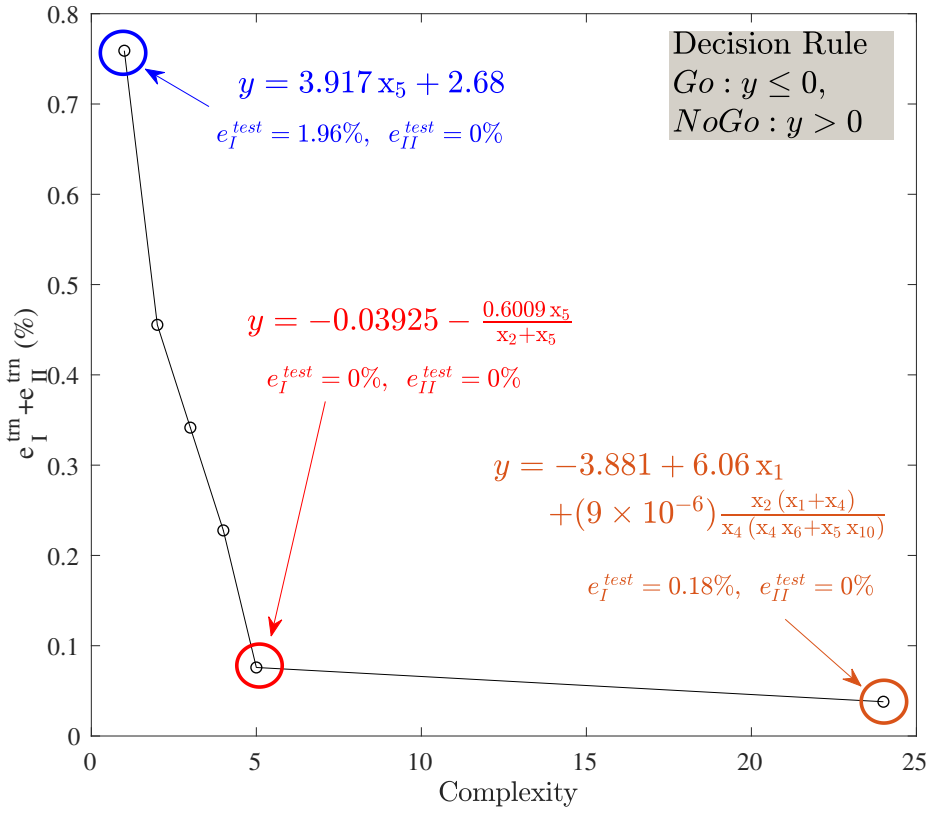


Figure 6.6: PO set of solutions found by CGP on solving the binary classification problem of Section 6.3.2.

set for the knee classifier discovered in data set-2 data are zero. One reason for such a result can be that these results are based on single runs of the CGP on the data. If multiple runs are made and classification errors are calculated by averaging the results of multiple runs, then we should at least be able to see a non-zero type-I error on the test set as has been the case with other results.

Table 6.6: Summary of classification rules found for binary classification problem of Section 6.3.2 and their corresponding error rates on training and test sets.

Soln Desc.	**Rule for *NoGo* products**	e_I^{trn}	e_{II}^{trn}	e_I^{test}	e_{II}^{test}
Simplest	$2.68 + 3.92x_5 > 0$	1.51%	0.00%	1.96%	0.00%
Knee	$-0.039 - \frac{0.6\,x_5}{x_2 + x_5} > 0$	0.15%	0.00%	0.00%	0.00%
Most Accurate	$-3.88 + 6.06x_1 + \frac{(9 \times 10^{-6})x_2(x_1 + x_4)}{x_4(x_4x_6 + x_5x_{10})} > 0$	0.08%	0.00%	0.18%	0.00%

6.4 Results on Larger Feature Space with Dimension Check

In the previous section, we presented some results showing the performance of CGP as a classifier on real production data, but with just ten features. After these initial encouraging results, our collaborators were much more interested in knowing

- How the CGP performs with a larger set of features. Furthermore,
- If a set of suitable constants for the process is provided, can dimensional analysis identify a set of dimensionally consistent classifiers from the PO set obtained from CGP?

6.4.1 Data and Results

Although, we conducted experiments on 5 years worth of production data ($>$ 6 Terabytes of data), here we are presenting the results of two successive days of production where first day's data is used as training set and the next day's data is used as test set. In this case, we

extracted a total of 56 features from the data, x_i where $i \in \{1, \ldots, 56\}$. These are shown in Table 6.7. We were also provided the units of measurement of those features. Furthermore,

Table 6.7: Larger feature set and their dimensions.

Feature(s)	Physical Dimension
x_1	$\mathrm{L}^0\ \mathrm{M}^0\ \mathrm{T}^1$
$x_2,\ x_3$	$\mathrm{L}^2\ \mathrm{M}^1\ \mathrm{T}^{-2}$
$x_i,\ i \in \{4, 5, \ldots, 13\}$	$\mathrm{L}^2\ \mathrm{M}^1\ \mathrm{T}^{-4}$
$x_i,\ i \in \{14, 15, \ldots, 23\}$	$\mathrm{L}^2\ \mathrm{M}^1\ \mathrm{T}^{-2}$
x_{24}	$\mathrm{L}^1\ \mathrm{M}^0\ \mathrm{T}^0$
$x_i,\ i \in \{25, 26, \ldots, 36\}$	$\mathrm{L}^1\ \mathrm{M}^0\ \mathrm{T}^{-1}$
$x_i,\ i \in \{37, 38, \ldots, 46\}$	$\mathrm{L}^1\ \mathrm{M}^0\ \mathrm{T}^1$
$x_i,\ i \in \{47, 48, \ldots, 56\}$	$\mathrm{L}^0\ \mathrm{M}^0\ \mathrm{T}^{-1}$

a list of four physical constants relevant to the physics of the process was also provided. These are listed in Table 6.8. Upon applying CGP as a classifier followed by the dimensional

Table 6.8: The set of physical constants which were considered relevant to the underlying physics of the production process.

Symbol	Name	Numeric Value	SI units	Physical Dimension
ρ	Material Density	2.7×10^3	kg/m^3	$\mathrm{L}^{-3}\ \mathrm{M}^1\ \mathrm{T}^0$
E	Young's Modulus	7.0×10^{10}	Pa	$\mathrm{L}^{-1}\ \mathrm{M}^1\ \mathrm{T}^{-2}$
H_v	Vickers Hardness	2.55×10^8	Pa	$\mathrm{L}^{-1}\ \mathrm{M}^1\ \mathrm{T}^{-2}$
τ	Material thickness	2.0×10^{-4}	m	$\mathrm{L}^1\ \mathrm{M}^0\ \mathrm{T}^0$

consistency check, the results of CGP are shown in Figure 6.7. Note that the classifiers found to be dimensionally inconsistent are marked in red. Table 6.9 shows the details of the 14 PO classifiers obtained by CGP for this case along with their type-I and type-II errors on the test set. Table 6.10 shows the classifiers shown in Table 6.9 after conducting the dimensional consistency check.

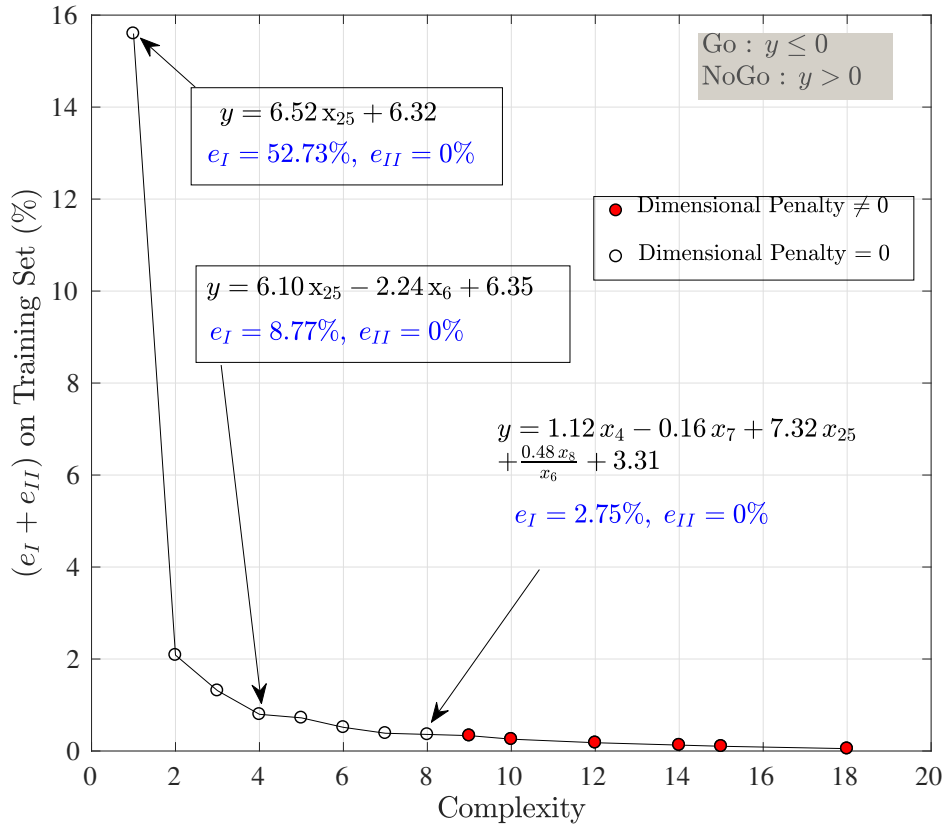


Figure 6.7: PO set of classifiers found by CGP for two day's worth production data described in Section 6.4.1.

6.5 Concluding Remarks

In this chapter we showed how can we apply CGP followed by dimensional analysis to learn simple and interpretable rules binary data. We could obtain dimensionally consistent rules from real world production data from industry. Such models bring valuable insight about the underlying physics of the process and can be used for both, for validating existing analytical laws obtained known from data as well as aid in building analytical models for physical processes which are not yet completely understood by researchers.

This concludes Part-I of this thesis in which we presented various ways in which we can learn simple and interpretable rules from data. Now, we move on to the other important aspect of online innovization, i.e. using this derived knowledge to intervene during an EMO

Table 6.9: Details of PO classifiers shown in Figure 6.7.

Soln ID	Expression ($NoGo{:=}y > 0$)	e_I^{test} (%)	e_{II}^{test} (%)
1	$y = 2.531\ (\mathrm{x}_4 + \mathrm{x}_{12})\ (\mathrm{x}_{25} + \mathrm{x}_{50}) - 0.2903\,\mathrm{x}_7 - 0.3398\,\mathrm{x}_{24} - 2.302\,\mathrm{x}_6 - \frac{1.142\,\mathrm{x}_5}{\mathrm{x}_3\,\mathrm{x}_4} + 0.7842$	0.7	0.0
2	$y = 2.446\ (\mathrm{x}_4 + \mathrm{x}_{12})\ (\mathrm{x}_{25} + \mathrm{x}_{50}) - 0.3075\,\mathrm{x}_7 - 2.446\,\mathrm{x}_6 - \frac{1.482\,\mathrm{x}_5}{\mathrm{x}_3\,\mathrm{x}_4} + 0.9329$	0.9	0.0
3	$y = 0.7485\,\mathrm{x}_4 - 2.999\,\mathrm{x}_6 - 0.4648\,\mathrm{x}_7 - \frac{0.6235\,\mathrm{x}_8}{\mathrm{x}_{25}} + 2.999\ (\mathrm{x}_4 + \mathrm{x}_{12})\ (\mathrm{x}_{25} + \mathrm{x}_{50}) - 1.432$	1.4	0.0
4	$y = 0.4964\,\mathrm{x}_4 - 1.465\,\mathrm{x}_5 - 2.401\,\mathrm{x}_6 - 0.2713\,\mathrm{x}_7 + 2.401\ (\mathrm{x}_4 + \mathrm{x}_{12})\ (\mathrm{x}_{25} + \mathrm{x}_{50}) + 0.5066$	1.0	0.0
5	$y = 0.4421\,\mathrm{x}_4 - 1.301\,\mathrm{x}_5 - 1.803\,\mathrm{x}_6 + 3.468\,\mathrm{x}_{25} + 2.917\,\mathrm{x}_{45} - \frac{0.5354\,\mathrm{x}_8}{\mathrm{x}_{25}} + 0.0814$	1.9	0.0
6	$y = 0.7815\,\mathrm{x}_4 - 2.063\,\mathrm{x}_6 + 3.934\,\mathrm{x}_{25} + 4.339\,\mathrm{x}_{45} - \frac{0.8135\,\mathrm{x}_8}{\mathrm{x}_{25}} - 2.6$	2.6	0.0
7	$y = 1.124\,\mathrm{x}_4 - 0.1635\,\mathrm{x}_7 + 7.317\,\mathrm{x}_{25} + \frac{0.4802\,\mathrm{x}_8}{\mathrm{x}_6} + 3.309$	2.8	0.0
8	$y = 0.9842\,\mathrm{x}_4 - 0.1513\,\mathrm{x}_7 + 6.852\,\mathrm{x}_{25} + \frac{0.4642\,\mathrm{x}_8}{\mathrm{x}_6} + 3.21$	3.2	0.0
9	$y = 4.768\,\mathrm{x}_{25} - 0.4487\,\mathrm{x}_7 - 1.274\,\mathrm{x}_5 + \frac{0.3278\,\mathrm{x}_4}{\mathrm{x}_6} + 4.121$	2.4	0.0
10	$y = 5.591\,\mathrm{x}_{25} - 1.631\,\mathrm{x}_5 + \frac{0.1578\,\mathrm{x}_8}{\mathrm{x}_6} + 5.252$	3.9	0.0
11	$y = 0.7103\,\mathrm{x}_4 - 1.761\,\mathrm{x}_6 - 1.342\,\mathrm{x}_7 + 4.998\,\mathrm{x}_{25} + 4.906$	3.5	0.0
12	$y = 4.894\,\mathrm{x}_{25} - 1.677\,\mathrm{x}_7 - 1.747\,\mathrm{x}_6 + 5.896$	4.2	0.0
13	$y = 6.097\,\mathrm{x}_{25} - 2.237\,\mathrm{x}_6 + 6.353$	8.8	0.0
14	$y = 6.516\,\mathrm{x}_{25} + 6.321$	52.7	0.0

run and hopefully expedite its convergence to the PO front. We will look at this aspect in the next part.

Table 6.10: Details of PO classifiers shown in Figure 6.7 after dimensional check.

ID	Original Expression ($NoGo{:=}y > 0$)	Dim. Penalty	Updated expression (with Physical constants of Table 6.8)
1	$y = 2.531\,(\mathrm{x}_4 + \mathrm{x}_{12})\,(\mathrm{x}_{25} + \mathrm{x}_{50}) - 0.2903\,\mathrm{x}_7 - 0.3398\,\mathrm{x}_{24} - 2.302\,\mathrm{x}_6 - \frac{1.142\,\mathrm{x}_5}{\mathrm{x}_3\,\mathrm{x}_4} + 0.7842$	2.309	NA
2	$y = 2.446\,(\mathrm{x}_4 + \mathrm{x}_{12})\,(\mathrm{x}_{25} + \mathrm{x}_{50}) - 0.3075\,\mathrm{x}_7 - 2.446\,\mathrm{x}_6 - \frac{1.482\,\mathrm{x}_5}{\mathrm{x}_3\,\mathrm{x}_4} + 0.9329$	2.309	NA
3	$y = 0.7485\,\mathrm{x}_4 - 2.999\,\mathrm{x}_6 - 0.4648\,\mathrm{x}_7 - \frac{0.6235\,\mathrm{x}_8}{\mathrm{x}_{25}} + 2.999\,(\mathrm{x}_4 + \mathrm{x}_{12})\,(\mathrm{x}_{25} + \mathrm{x}_{50}) - 1.432$	2.374	NA
4	$y = 0.4964\,\mathrm{x}_4 - 1.465\,\mathrm{x}_5 - 2.401\,\mathrm{x}_6 - 0.2713\,\mathrm{x}_7 + 2.401\,(\mathrm{x}_4 + \mathrm{x}_{12})\,(\mathrm{x}_{25} + \mathrm{x}_{50}) + 0.5066$	4.041	NA
5	$y = 0.4421\,\mathrm{x}_4 - 1.301\,\mathrm{x}_5 - 1.803\,\mathrm{x}_6 + 3.468\,\mathrm{x}_{25} + 2.917\,\mathrm{x}_{45} - \frac{0.5354\,\mathrm{x}_8}{\mathrm{x}_{25}} + 0.0814$	0.707	NA
6	$y = 0.7815\,\mathrm{x}_4 - 2.063\,\mathrm{x}_6 + 3.934\,\mathrm{x}_{25} + 4.339\,\mathrm{x}_{45} - \frac{0.8135\,\mathrm{x}_8}{\mathrm{x}_{25}} - 2.6$	0.707	NA
7	$y = 1.124\,\mathrm{x}_4 - 0.1635\,\mathrm{x}_7 + 7.317\,\mathrm{x}_{25} + \frac{0.4802\,\mathrm{x}_8}{\mathrm{x}_6} + 3.309$	0.0	$y = 1.48 \times 10^{12}\,\frac{\rho}{EH_v\tau}x_4 - 2.12 \times 10^{11}\,\frac{\rho}{EH_v\tau}x_7 + 2249.54\sqrt{\frac{\rho}{H_v}}x_{25} + \frac{0.48\,x_8}{x_6} + 3.31$
8	$y = 0.9842\,\mathrm{x}_4 - 0.1513\,\mathrm{x}_7 + 6.852\,\mathrm{x}_{25} + \frac{0.4642\,\mathrm{x}_8}{\mathrm{x}_6} + 3.21$	0.0	$y = 1.30 \times 10^{12}\,\frac{\rho}{EH_v\tau}\mathrm{x}_4 - 2.0 \times 10^{11}\,\frac{\rho}{EH_v\tau}\mathrm{x}_7 + 2105.69\sqrt{\frac{\rho}{H_v}}\mathrm{x}_{25} + \frac{0.4642\,\mathrm{x}_8}{\mathrm{x}_6} + 3.21$
9	$y = 4.768\,\mathrm{x}_{25} - 0.4487\,\mathrm{x}_7 - 1.274\,\mathrm{x}_5 + \frac{0.3278\,\mathrm{x}_4}{\mathrm{x}_6} + 4.121$	0.0	$y = 1465.25\sqrt{\frac{\rho}{H_v}}\mathrm{x}_{25} - 5.93 \times 10^{11}\,\frac{\rho}{EH_v\tau}\mathrm{x}_7 - 1.68 \times 10^{12}\,\frac{\rho}{EH_v\tau}\mathrm{x}_5 + \frac{0.3278\,\mathrm{x}_4}{\mathrm{x}_6} + 4.121$
10	$y = 5.591\,\mathrm{x}_{25} - 1.631\,\mathrm{x}_5 + \frac{0.1578\,\mathrm{x}_8}{\mathrm{x}_6} + 5.252$	0.0	$y = 1718.17\sqrt{\frac{\rho}{H_v}}\mathrm{x}_{25} - 2.16 \times 10^{12}\,\frac{\rho}{EH_v\tau}\mathrm{x}_5 + \frac{0.1578\,\mathrm{x}_8}{\mathrm{x}_6} + 5.252$
11	$y = 0.7103\,\mathrm{x}_4 - 1.761\,\mathrm{x}_6 - 1.342\,\mathrm{x}_7 + 4.998\,\mathrm{x}_{25} + 4.906$	0.0	$y = 9.39 \times 10^{11}\,\frac{\rho}{EH_v\tau}\mathrm{x}_4 - 2.33 \times 10^{12}\,\frac{\rho}{EH_v\tau}\mathrm{x}_6 - 1.77 \times 10^{12}\,\frac{\rho}{EH_v\tau}\mathrm{x}_7 + 1535.93\sqrt{\frac{\rho}{H_v}}\mathrm{x}_{25} + 4.906$
12	$y = 4.894\,\mathrm{x}_{25} - 1.677\,\mathrm{x}_7 - 1.747\,\mathrm{x}_6 + 5.896$	0.0	$y = 1503.97\sqrt{\frac{\rho}{H_v}}\mathrm{x}_{25} - 1.68 \times 10^{12}\,\frac{\rho}{EH_v\tau}\mathrm{x}_7 - 1.747\,\mathrm{x}_6 + 5.896$
13	$y = 6.097\,\mathrm{x}_{25} - 2.237\,\mathrm{x}_6 + 6.353$	0.0	$y = 1874.62\sqrt{\frac{\rho}{H_v}}\mathrm{x}_{25} - 2.96 \times 10^{12}\,\frac{\rho}{EH_v\tau}\mathrm{x}_6 + 6.35$
14	$y = 6.516\,\mathrm{x}_{25} + 6.321$	0.0	$y = 2003.69\sqrt{\frac{\rho}{H_v}}\mathrm{x}_{25} + 6.32$

Part II

Rule Based Repair

Chapter 7

Performing Repairs Based on Fixed Form Rules for Expedited Convergence

Part I of this dissertation has discussed ways in which we can distill fixed and free form rules or mathematical expressions from data. Recall from Figure 2.6 that when we use an EMO algorithm to solve a MOO problem, applying an online innovization procedure involves:

- First choosing a set of solutions to learn rules from, then
- Learning rules or patterns that exist in those solutions and then,
- Perform an intervention in the EMO algorithm to expedite its convergence rate.

The two chapters in Part II will focus on using learned rules of fixed form discussed in Chapter 3 to intervene in the EMO algorithm and possibly expedite its convergence towards the PO front of MOO problem.

7.1 Rule Based Repair

Recall from Section 3.1 that *basis function(s)* for a rule in the context of innovization can be any scalar function ϕ of the design variables including variables, objectives, constraints

and any function thereof. Fixed form rules of constant type, given by Equation (3.4), and of power-law form, given by Equation (3.7), can be represented in a single equation using the form given by Equation (7.1) as

$$\begin{aligned} &\psi_i(\phi_1, \phi_2, \ldots, \phi_{n_\phi}) \equiv \prod_j^{n_\phi} \phi_j^{a_{ij}\, b_{ij}} = c_i, \text{ where,} \\ &a_{ij} \in \{0, 1\}\ ,\ c_i, b_{ij} \in \mathbb{R},\ n_\phi = \#\text{ of basis functions.} \end{aligned} \tag{7.1}$$

In Equation (7.1), ψ_i represents i^{th} candidate rule, ϕ_j represents the j^{th} basis function in the list of n_ϕ basis functions being considered for learning rules, a_{ij} is a binary variable that decides if j^{th} basis function is included in i^{th} candidate rule ψ_i and, b_{ij}, c_i are rule parameters that are estimated during an EMO/I run. If ψ_i is a constant rule, say $\phi_j = c_i$ then

$$a_{ij} = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{7.2}$$

This chapter considers candidate rules involving only design variables as basis functions, thus Equation (7.1) takes the form

$$\psi_i(x_1, x_2, \ldots, n_x) \equiv \prod_{j=1}^{n_x} x_j^{a_{ij} \cdot b_{ij}} = c_i \tag{7.3}$$

where n_x is the number of design variables in a MOO problem. We refer to the set of rules represented by Equation (7.3) as *candidate rules*. In a problem with n_x design variables, a maximum of $\sum_{k=1}^{n_x} {}^{n_x}C_k = 2^{n_x} - 1$ candidate rules of fixed form involving just variables as basis functions can be evaluated. Out of those, n_x rules are constant type rules corresponding to each variable and the rest are power law type of rules.

Figure 7.1 shows a flowchart of the online innovization applied to EMO algorithms or "EMO/I" method proposed in Section 2.5. Except for the three regular blocks, namely *the Rule Basis and Quality* block, *Learn* block and *Repair* block and two decision blocks, namely L and R blocks, the rest of the flowchart is that of a generic EMO algorithm. In this work, we used NSGA-II algorithm [92] as the EMO algorithm of choice but it can be any population based optimization method. The following sections provide a brief description of

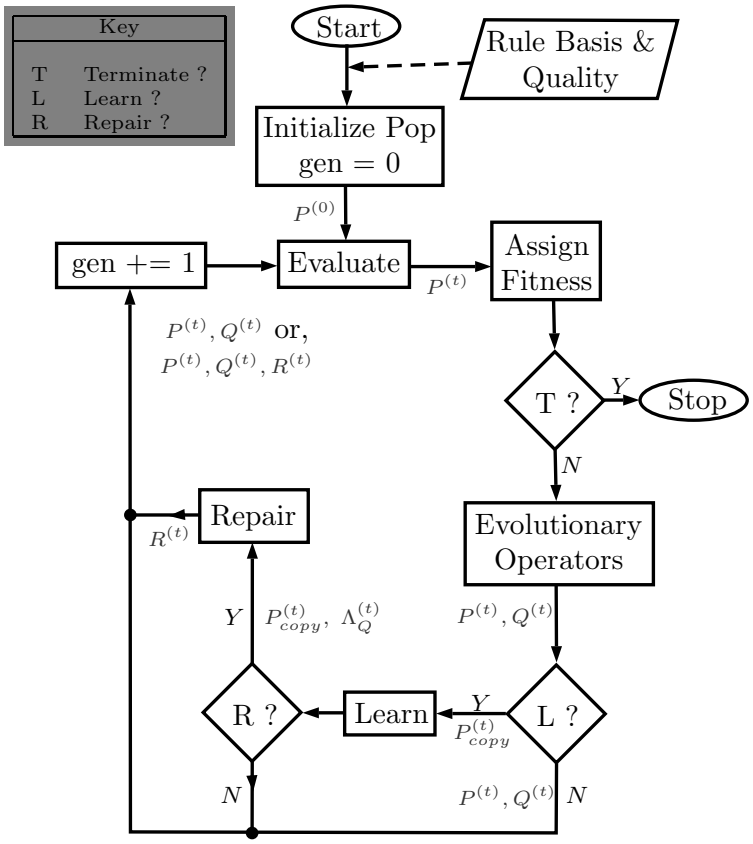


Figure 7.1: The flowchart of an EMO/I algorithm.

the flowchart with a focus on the aforementioned blocks that are specific to EMO/I.

7.1.1 Rule Basis and Quality Block

The *rule basis and quality* or RBQ block carries information on:

1. Which rules out of the maximum of $2^{n_x} - 1$ that we wish to learn? and,

2. What is the threshold quality measure on a candidate rule to consider it suitable for variable repair?

Information for both of these questions is sought from the user at the beginning of EMO/I procedure. For example, if a MOO problem has three variables but if the user wants to discover rules with a maximum two variable interactions, then we can set a_{ij} as shown in Table 7.1.

Table 7.1: An example of candidate rule information available in RBQ block.

Rule Id (i)	a_{ij} values			Rule type
1	$a_{11} = 1$	$a_{12} = 0$	$a_{13} = 0$	Constant rule
2	$a_{21} = 0$	$a_{22} = 1$	$a_{23} = 0$	Constant rule
3	$a_{31} = 0$	$a_{32} = 0$	$a_{33} = 1$	Constant rule
4	$a_{41} = 1$	$a_{42} = 1$	$a_{43} = 0$	Power law rule
5	$a_{51} = 1$	$a_{52} = 0$	$a_{53} = 1$	Power law rule
6	$a_{61} = 0$	$a_{62} = 1$	$a_{63} = 1$	Power law rule

Each candidate rule is learned and evaluated for its quality. For a constant rule, this quality is measured using the coefficient of variation (C_v) metric (see Section 3.2). A constant rule is considered good quality and ready for making repairs if its $C_v \leq C_v^{(max)}$. Similarly for power-law type of rules, we use the R^2_{adj} metric from OLSR to assess its quality (see Section 3.3). A power-law type of rule is considered good quality and ready for making repairs if its $R^2_{adj} \geq R^{2\,(min)}_{adj}$. We have kept the default value of these thresholds as $C_v^{(max)} = 0.05$ and $R^{2\,(min)}_{adj} = 0.95$. The user may change these values as per his/her requirement.

A Note on Transition Points

A *transition point* in a bi-objective PO front is a point across which the nature of mathematical relations among the PO solutions changes. The innovized principles on either side

of the transition point are different. Some reasons for encountering transition points are:

(a) Some constraint or variable bound becomes active (or inactive) and forces (or eases) the PO solutions to adhere to additional (or fewer) rules across the transition boundary,

(b) The nature of one or more of the objective functions changes significantly across the transition boundary.

In this chapter, we will consider optimization MOO problems that do not have a transition point and we will come back to addressing it in Chapter 8. Thus for problems in this chapter, the rules of the form given by Equation (7.1), if discovered, are applicable to all the PO solutions.

7.1.2 Decision Block-L

Once the EMO/I begins and a population of individuals is initialized, evaluated, assigned fitness and operated by genetic operators (crossover and mutation), the decision block 'L' decides how long should the algorithm wait before it begins to learn the parameters and quality of candidate rules. This is necessary as in the initial phases of an EMO algorithm, the solutions may still be far away from the PO-front and not adhere to any rule. Such a *waiting* decision can be implemented in many ways such as:

- If some minimum fraction of maximum function evaluations (MFEs) allowed for a problem have passed or,
- If the number of solutions in the non-dominated set is above a minimum threshold or,
- If the increase in hypervolume has slowed down to some pre-specified level.

In this work, we have gone ahead with using the first one, i.e. some fraction of MFEs, to be the deciding criterion to start learning rules.

7.1.3 Learn Block

This block attempts to learn the rule parameters b_{ij} and c_i shown in Equation (7.3), as well as the quality of each candidate rule. Once EMO/I meets the criterion set by the decision block 'L', a copy of the parent population, $P^{(t)}_{copy}$, is sent to the 'Learn' block. The rule parameters and the rule quality for constant rules and power-law rules are estimated as follows :

7.1.3.1 Constant Type Rules

Consider a rule of the form given in Equation (7.3) and a MOO problem with n_x variables. Then, for a candidate rule ψ_i composed of j^{th} variable x_j, the constant rule is given by

$$\psi_i(\phi_j) \equiv x_j = c_i. \tag{7.4}$$

Comparing with Equation (7.3), we know in this case that

$$\begin{aligned} a_{ik} &= 1 \text{ if } k = j, \text{ 0 otherwise,} \\ b_{ik} &= 1, \text{ if } k = j, \text{ 0 otherwise,} \end{aligned} \tag{7.5}$$

where $k \in \{1, 2, \ldots, n_x\}$. From Section 3.2.1, we know that the value of parameter c_i can be estimated as

$$\hat{c}_i = \mu_j \tag{7.6}$$

where μ_j is the mean of j^{th} decision variable over the non-dominated solutions set

Furthermore, the *coefficient of variation* (C_v), of the j^{th} variable over the non-dominated solutions set is considered as a measure of the quality for such a rule. This quality parameter is later used in the decision block R shown in Figure 7.1.

7.1.3.2 Power Law Type of Rules

As shown in Section 3.3.1, to estimate the parameters and quality of power law rules, we use log-linear modeling followed by applying OLSR on data of non-dominated solutions to learn the parameters.Consider an example of a three variable MOO problem, and we want to learn the parameters of a two variable power law rule as:

$$x_1^{b_1} x_3^{b_3} = c, \text{ then} \tag{7.7}$$

taking log on both sides,

$$b_1 \log x_1 + b_3 \log x_3 = \log c,$$

which is a linear equation. If $\log x_1$ is chosen as the regressand then,

$$\begin{aligned} \log x_1 &= \frac{-b_3}{b_1} \log x_3 + \frac{\log c}{b_1}, \text{ or} \\ &= \hat{\beta} \log x_3 + \hat{\gamma}, \\ \implies x_1 &= e^{\hat{\gamma}} x_3^{\hat{\beta}} \end{aligned} \tag{7.8}$$

Parameters $\hat{\beta}$ and $\hat{\gamma}$ are estimates returned by a *Ordinary Least Square* linear regression (OLSR) method using the log of 1^{st} and 3^{rd} variables from non-dominated set. OLSR also returns the R^2_{adj} value which is later used to assess the quality of such a candidate rule in

repair stage.

7.1.4 Decision Block-R

This block decides if some rule is good enough in quality to qualify for the repair stage. As mentioned in Section 7.1.1,

- The quality of one variable rules is measured using coefficient of variation (C_v) values and,
- The quality of rules with more than one variable is measured using R^2_{adj} value returned from OLSR.

EMO/I is provided with threshold quality parameters namely, the maximum coefficient of variation ($C_v^{(max)}$) for constant type rules and $R^{2\,(min)}$ for power law rules, in the RBQ block at the start of EMO/I. A rule is said to qualify for repair in next stage as follows:

- For constant rules, $C_v \leq C_v^{(max)}$ and
- For power law rules, $R^2_{adj} \geq R^{2\,(min)}_{adj}$.

We refer to the candidate rules for which the quality parameters surpass the threshold quality parameter values as *Qualifying Rules*. As set of u such qualifying rules in generation t of EMO/I, say $\Psi_Q^{(t)} = \{\psi_{q_1}, \psi_{q_2}, \ldots, \ldots, \psi_{q_u}\}$, along with a copy of parent population, say $P_{copy}^{(t)}$, is passed on to the *Repair* block as shown in Figure 7.1. In this work, we refer to the union of set of variables constituting the qualifying rules as *Qualifying Variables*.

7.1.5 Repair Block

In some generation t of EMO/I, the repair block receives a copy of parent population ($P^{(t)}_{copy}$) and a set of qualifying rules ($\Psi^{(t)}_{Q} = \{\psi_{q_1}, \psi_{q_2}, \ldots, \ldots, \psi_{q_u}\}$), and yields a population with repaired variables ($R^{(t)}$). The procedure for making such a repair is given by Algorithm 7.1 and is named RepairPop( ). Next, we briefly describe the overall algorithm and followed by detailed explanation its important components.

Algorithm 7.1 RepairPop( )

input: $P^{(t)}_{copy}$, $\Psi^{(t)}_{Q} = \{\psi_{q_1}, \psi_{q_2}, \ldots, \ldots, \psi_{q_u}\}$ Parent population, List of Qualifying rules
output: $R^{(t)}$ Repaired population

1: $R^{(t)} \leftarrow \emptyset$
2: **for** $k \leftarrow 1$ **to** $|P^{(t)}_{copy}|$ **do**
3: $P_k \leftarrow k^{th}$ individual of $P^{(t)}_{copy}$
4: $\mathcal{I} \leftarrow \emptyset$ Initialize set of repaired variables in P_k.
5: $\tilde{\Psi}^{(t)}_{Q} \leftarrow$ wrShuffle($\Psi^{(t)}_{Q}$) Wtd. random shuffle w.r.t rule length preference.
6: **for** $m \leftarrow 1$ **to** $|\tilde{\Psi}^{(t)}_{Q}|$ **do**
7: $\psi \leftarrow m^{th}$ rule of $\tilde{\Psi}^{(t)}_{Q}$
8: $\mathcal{I}_{\psi} \leftarrow$ getVars(ψ) Get set of variables in rule ψ.
9: $\mathcal{J} \leftarrow \mathcal{I}_{\psi} \setminus \mathcal{I}$ Get set of variables in rule ψ and not yet repaired in P_k.
10: **if** $\mathcal{J} \neq \emptyset$ **then**
11: $v \leftarrow$ chooseVar($\mathcal{J}$) Get variable w.r.t frequency preference.
12: makeRepair(P_k, v) Repair variable v of individual P_k.
13: $\mathcal{I} \leftarrow \mathcal{I} \cup v$
14: $R^{(t)} \leftarrow R^{(t)} \cup P_k$ Update repaired population set.

RepairPop( ) procedure takes $P^{(t)copy}$ and $\Lambda^{(t)}_{Q}$ as input and further line wise description is as below:

- Lines 1-4 : The output $R^{(t)}$ is initialized as an empty set. It then loops over all individuals of $P^{(t)}_{copy}$ to repair them one by one. For the k^{th} individual of $P^{(t)}_{copy}$, P_k, the set of repaired variables in P_k is initialized to an empty set.

- Lines 5-6 : The rules present in $\Psi^{(t)}_{Q}$ are shuffled using wrShuffle() function with

respect to some probability distribution based on *length of rules*. This is discussed in Section 7.2.1. We refer to the number of variables in a rule as the length of a rule. For example, the rule $\lambda \equiv x_1 \cdot x_2 = 3$ has length two. The procedure then sequentially goes over all rules in the sorted list $\tilde{\Psi}_Q^{(t)}$.

- Lines 7-9 : For some m^{th} rule in the shuffled rule list $\tilde{\Psi}_Q^{(t)}$, say ψ, the set of variables involved in rule ψ is stored in $\mathcal{I}_\psi$ using GETVARS() function. For example, for the rule $\psi \equiv x_1 \cdot x_2 = 3$, $\mathcal{I}_\psi = \{x_2, x_3\}$. Subsequently, set of variables that are involved in rule ψ and have not been repaired in the individual P_k is stored in $\mathcal{J}$.

- Lines 10-13 : If $\mathcal{J} = \emptyset$, then the control passes back to the **for** loop of line-6. Else, some variable $v \in \mathcal{J}$ is chosen using CHOOSEVAR() function. This choice depends on a probability distribution based on *frequency* of the variables in the qualifying rules and is explained in Section 7.2.2. Subsequently, the variable v is repaired in individual P_k using MAKEREPAIR() function and its operation is explained in Sections 7.1.5.1 and 7.1.5.2. After repairing variable v, the set $\mathcal{I}$ is updated.

- Line 14 : Once all possible repairs have been performed to the individual P_k, it is added to the repaired population $R^{(t)}$. This continues until **for** loop of line-2 covers all individuals of $P_{copy}^{(t)}$.

Next we discuss the working of the MAKEREPAIR() function of Algorithm 7.1.

7.1.5.1 Repairing Variables Based on Constant Rule

Consider a rule of the form given in Equation (7.4) corresponding to a candidate rule Ψ_i composed of j^{th} variable x_j.

$$\Psi_i \equiv x_j = c_i, \text{ where} \tag{7.9}$$

c_i is as estimated as given in Equation (7.6). Then, the j^{th} variable in an individual of $P_{copy}^{(t)}$ population is repaired as,

$$\hat{x}_j = \mathcal{U}(\mu_j - \sigma_j, \mu_j + \sigma_j), \text{ where} \tag{7.10}$$

$\hat{x}_J$ is the repaired value of variable x_j, μ_j and σ_j are the mean and standard-deviation respectively of x_j decision variable over the non-dominated solutions set and $\mathcal{U}(a, b)$ represents a uniform random distribution between a and b.

7.1.5.2 Repairing variables based on power law rules

To explain the repair method involved in this case, we take the example shown in Section 7.1.3.2. From Equation (7.8), we can get the repaired value of variable x_1 as

$$\hat{x}_1 = e^{\hat{\gamma}} x_3^{\hat{\beta}}. \tag{7.11}$$

Similarly, the repaired value for the regressor variable x_3 can be obtained as

$$\hat{x}_3 = \left(\frac{x_1}{e^{\hat{\gamma}}}\right)^{\frac{1}{\hat{\beta}}} \tag{7.12}$$

For cases with more than two variables as well, a similar logic follows. Note that if the repaired value of a variable lies outside its a priori defined bounds, then the repaired variable is set to its nearest bound value.

7.2 Repair Strategies

This section discusses three *rule-preference* strategies, corresponding to the wrShuffle() function, and three *variable-preference* strategies corresponding to the chooseVar() function, used in Algorithm 7.1.

Recall that we call a set of rules for which the quality parameters surpass the threshold value of quality set in RBQ block of Figure 7.1 as qualifying rules, and the union of set of variables constituting these qualifying rules as qualifying variables. The following two choices need to be made before any repair of the kinds illustrated in Section 7.1.5 is made to solution individuals of $P_{copy}^{(t)}$:

- Choose one of the possibly many qualifying rules on which to base the repair and,
- Choose one of the possibly many variables for repair from the chosen qualifying rule.

For example, choosing a random rule from the qualifying rules pool and then choosing a random variable from the variables of the chosen qualifying rule can be one such strategy.Both these choices have an effect on:

1. The parameters used to make a repair and,
2. The sequence in which variables are repaired.

7.2.1 Rule Preference Strategies

We investigate three rule-preference strategies. We call the number of variables in a rule to be the *length of that rule*. Before discussing these strategies, lets look at the wrShuffle() function as it is used in implementing all three rule preference strategies in Line-5 of Algorithm 7.1.

The WRSHUFFLE() function stands for *weighted random shuffle* (WRS) and it is same as weighted random sampling from a sequence without replacement. Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ be a sequences of n objects and unnormalized weights respectively such that weight w_i corresponds to element a_i. Then Algorithm 7.2 describes a pseudo code for WRSHUFFLE() function.

Algorithm 7.2 WRSHUFFLE( )

input: $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$
output: $\boldsymbol{r}$ — Wtd. Random Shuffled sequence

1: $\boldsymbol{r} \leftarrow \emptyset$
2: $n \leftarrow$ SIZEOF($\boldsymbol{a}$) — get number of elements in sequence $\boldsymbol{a}$
3: **while** $n > 0$ **do**
4: $\quad s \leftarrow \sum_{k=1}^{n} w_k$
5: $\quad \mathcal{P} \leftarrow (p_1, p_2, \ldots, p_n)$ where $p_i = w_i/s$ — Probability mass value sequence.
6: $\quad \mathcal{C} \leftarrow (0, \sum_{k=1}^{1} p_k, \sum_{k=1}^{2} p_k, \ldots, \sum_{k=1}^{n} p_k)$ — Cumulative distribution value sequence.
7: $\quad r \leftarrow \mathcal{U}(0, 1)$ — Random no. in 0-1 using uniform random distribution.
8: $\quad i \leftarrow 1$
9: $\quad$ **while** true **do**
10: $\quad\quad$ **if** $\mathcal{C}_i \leq r < \mathcal{C}_{i+1}$ **then** — $\mathcal{C}_i$ is i^{th} element of $\mathcal{C}$
11: $\quad\quad\quad$ break
12: $\quad\quad$ **else**
13: $\quad\quad\quad i \leftarrow i + 1$
14: $\quad \boldsymbol{r} \leftarrow \boldsymbol{r}^{\frown} < a_i >$ — Append a_i to sequence $\boldsymbol{r}$ at end.
15: $\quad \boldsymbol{a} \leftarrow \boldsymbol{a} \backslash < a_i >$ — Remove element a_i from sequence $\boldsymbol{a}$.
16: $\quad n \leftarrow n - 1$

Lets look at the rule preference strategies next. Let there be m rules in the qualifying rules list $\Lambda_Q^{(t)} = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ of Algorithm 7.1 with lengths $(l_1, l_2, \ldots, l_m)$ respectively. Then the three rule-preference strategies assign different weights sequences to the m qualifying rules in WRSHUFFLE() function.

(i) *No preference*: In this strategy, no rule is given preference over others based on its length and a weights sequence of $\boldsymbol{w} = (1, 1, \ldots, 1)$ is used by WRSHUFFLE() function.

(ii) *Prefer long rules*: In this strategy, lengthier rules are preferred over shorter rules and

a weights sequence of $\boldsymbol{w} = (l_1, l_2, \ldots, l_m)$ is used by WRSHUFFLE() function.

(iii) *Prefer short rules*: In this strategy, shorter rules are preferred over lengthier rules and a weights sequence of $\boldsymbol{w} = (1/l_1, 1/l_2, \ldots, 1/l_m)$ is used by WRSHUFFLE() function.

7.2.2 Variable Preference Strategies

We investigate three variable-preference strategies which are based on the frequency of a variable among the qualifying rules and it is implemented using the CHOOSEVAR() function in Algorithm 7.1. In Algorithm 7.1, let there be n qualifying variables, $\mathcal{V}_Q^{(t)} = \{v_1, v_2, \ldots, v_n\}$, in the list of qualifying rules $\Psi_Q^{(t)} = (\psi_1, \psi_2, \ldots, \psi_m)$. Furthermore, let the frequency of each qualifying variable in $\mathcal{V}_Q^{(t)}$ be represented by $\mathcal{F}_Q^{(t)} = \{f_1, f_2, \ldots, f_n\}$ where f_i is frequency of qualifying variable v_i. Let some rule $\psi_i \in \Psi_Q^{(t)}$ be under consideration for making variable repair to population individual P_k at some step of **for** loop of line-6 of Algorithm 7.1. Then, $\mathcal{J} \subseteq \mathcal{V}_Q^{(t)}$, represents the set of variables in rule λ_i that have not been repaired in the individual P_k until that instant. Let,

$$\begin{aligned} \mathcal{J} &= \{v_{j_1}, v_{j_2}, \ldots, v_{j_s}\}, \text{ and} \\ \mathcal{F} &= \{f_{j_1}, f_{j_2}, \ldots, f_{j_s}\}, \text{ where} \end{aligned} \tag{7.13}$$

$f_{j_u} \in \mathcal{F}$ is the frequency of variable $v_{j_u} \in \mathcal{J}$ in the qualifying variable set $\mathcal{V}_Q^{(t)}$ and $u \in \{1, 2, \ldots, s\}$. Let us discuss the variable-preference strategies.

(i) *No preference*: In this strategy, no variable is given preference over others based on its frequency among the qualifying variables. In this case, the CHOOSEVAR() function randomly picks one variable from the set $\mathcal{J}$ defined in Equation (7.13).

(ii) *Prefer common variables*: In this strategy, qualifying variables that are more frequent among the qualifying variables are given a higher preference of getting selected first to get repaired. In this case, the CHOOSEVAR() function picks some variable $v_{ju} \in \mathcal{J}$ of Equation (7.13) with probability p_u defined as;

$$p_u = \frac{f_{ju}}{\sum_{r=1}^{s} f_{jr}} \tag{7.14}$$

(iii) *Prefer less-common variables*: In this strategy, qualifying variables that are less frequent among the qualifying variables are given a higher preference of getting selected first to get repaired. In this case, the CHOOSEVAR() function picks some variable $v_{ju} \in \mathcal{J}$ of Equation (7.13) with probability p_u defined as;

$$p_u = \frac{1/f_{j_u}}{\sum_{r=1}^{s} 1/f_{j_u}} \tag{7.15}$$

These aforementioned strategies in Sections 7.2.1 and 7.2.2, when permuted together form a total of nine strategy combinations. These are listed in the first nine rows of Table 7.2. The tenth strategy is of a pure EMO algorithm without any innovization based repairs.

Table 7.2: Different variable repair strategies for EMO/I studied in this work.

ID	Rules Preference	Variable Preference	Abbrev.
1	None	None	NN
2	None	Common variables	NC
3	None	Uncommon variables	NU
4	Long rules	None	LN
5	Long rules	Common variables	LC
6	Long rules	Uncommon variables	LU
7	Short rules	None	SN
8	Short rules	Common variables	SC
9	Short rules	Uncommon variables	SU
10	——— NSGA-II with No Repair ———-		NI

7.3 Test Problems

All test problems in this work have been derived from the ZDT1 problem [107] and have the following form.

$$\begin{aligned} \text{Minimize} \quad & f_1(\mathbf{x}) = x_1, \\ & f_2(\mathbf{x}) = g(\mathbf{x})\, h(f_1(\mathbf{x}), g(\mathbf{x})), \\ \text{Where} \quad & h(f_1, g) = 1 - \sqrt{f_1/g}. \end{aligned} \tag{7.16}$$

Every problem has a different g function, variable bounds and Pareto-optimal set and they are described in the following sections.

7.3.1 ZDT1-1

This problem is designed to have rules in the PO set such that:

- No variable is common among any two rules and,
- Different rules may have different number of variables.

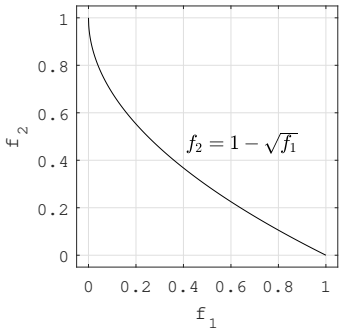
The problem is given by Equations (7.16) where $g(\mathbf{x})$ is given by Equation (7.17).

$$\begin{aligned} &g(\mathbf{x}) = 1 + |x_2 - 0.5| + |x_3 x_4 - 0.5| + |x_5 x_6 x_7 - 0.5|, \\ &x_{1,2} \in [0, 1],\ x_{3,4,5,6} \in [0.5, 1],\ x_7 \in [0.5, 2]. \end{aligned} \tag{7.17}$$

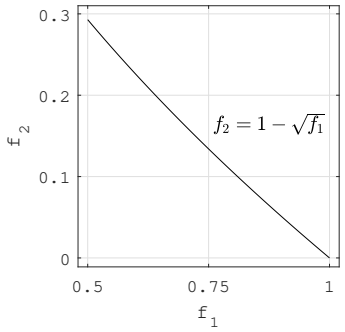
Equation (7.18) shows the PO solutions set for this problem and Figure 7.2a shows the corresponding PO front.

$$0.0 \leq x_1^* \leq 1.0,\ x_2^* = 0.5,\ x_3^* \cdot x_4^* = 0.5,\ \text{and},\ x_3^* \cdot x_4^* \cdot x_7^* = 0.5. \tag{7.18}$$

This problem is specifically designed to test repair strategies: NI, NN, SN and LN. Refer to



(a) Pareto-optimal front for ZDT1-1 problem.



(b) Pareto-optimal front for ZDT1-2 and ZDT1-3 problems.

Figure 7.2: Pareto-optimal fronts for the test problems.

Table 7.2 for description of these strategies.

7.3.2 ZDT1-2

This problem is designed to have rules in the PO set such that

- Some variables are common among two or more rules and,
- Each rule has the same number of variables.

The ZDT1-2 problem is given by Equation (7.16) where $g(\mathbf{x})$ is given by Equation (7.19).

$$\begin{aligned} &g(\mathbf{x}) = 1 + |x_1 x_2 - 0.5| + |x_2^{0.5} x_3 - \sqrt{0.5}| + |x_2^2 x_4 - 0.25| + |x_2^{-0.5} x_5 - \sqrt{0.5}|, \\ &x_{1,2} \in [0.5, 1],\ x_3 \in [\sqrt{0.5}, 1],\ x_4 \in [0.25, 1], x_5 \in [0.5, \sqrt{0.5}]. \end{aligned} \tag{7.19}$$

The Pareto-optimal region for this problem is given by Equation (7.20) and Figure 7.2b shows the corresponding PO front.

$$x_1^* \cdot x_2^* = 0.5,\ (x_2^*)^{0.5} \cdot x_3^* = \sqrt{0.5},\ (x_2^*)^2 \cdot x_4^* = 0.25 \text{ and, } (x_2^*)^{-0.5} \cdot x_5^* = \sqrt{0.5}. \tag{7.20}$$

This problem is designed to test repair strategies: NI, NN, NC and NU, shown in Table 7.2.

7.3.3 ZDT1-3

This problem is designed to have rules in the PO set such that:

- Some variables may be common among two or more rules and,
- Each rule may have different number of variables.

The ZDT1-3 problem is given by Equation (7.16) where $g(\mathbf{x})$ is given by Equation (7.21).

$$\begin{aligned} &g(\mathbf{x}) = 1 + |x_1 x_2 - 0.5| + |x_1^{0.5} x_2 x_3 - 0.5| + |x_5 - 0.5| + |x_1^{-0.5} x_2^2 x_3 x_4 - 0.25|, \\ &x_{1,2} \in [0.5, 1],\ x_3 \in [\sqrt{0.5}, 1],\ x_4 \in [0.25, 1],\ x_5 \in [0, 1]. \end{aligned} \tag{7.21}$$

The Pareto-optimal region for this problem is given by Equation (7.22) and Figure 7.2b shows the corresponding PO front.

$$x_1^* \cdot x_2^* = 0.5,\ (x_1^*)^{0.5} \cdot x_2^* \cdot x_3^* = 0.5,\ x_5^* = 0.5 \text{ and, } (x_1^*)^{-0.5} \cdot (x_2^*)^2 \cdot x_3^* \cdot x_4^* = 0.25\ . \tag{7.22}$$

This problem is designed to test and compare all the repair strategies given in Table 7.2.

7.4 Results on Test Problems

This section compares the performance of an EMO algorithm, NSGA-II in this work, with that of an EMO/I algorithm using the different variable repair strategies of Table 7.2 on the test problems of Section 7.3. The tables and figures in this section refer to different contending algorithms by their respective abbreviations mentioned in Table 7.2. For example, an EMO algorithm without any variable repair strategy is referred to as 'NI'.

Furthermore, the different algorithms are compared on the metrics of median *Generational Distance* (GD) and median *Inverse Generational Distance* (IGD) [31] over thirty runs. All claims of one algorithm being better than the other are backed with results of *Wilcoxon Rank Sum* (WRS) test [108] of statistical significance. We use the following convention to represent a test hypothesis. $\mathcal{H}_{\mathrm{A}\prec\mathrm{B}}$ *represents the left tailed hypothesis test, where the alternative hypothesis states that the median of distribution A is lower than the median of distribution B at some significance level* α. A significance level of $\alpha = 5\%$ is used in all the statistical tests. For all the statistical tests, both h and p values are shown in the results. An h-value of 'No' means that the aforementioned alternative hypothesis cannot be accepted at the desired significance level and an h-value of 'Yes' means otherwise. Also, for the aforementioned alternative hypothesis to be accepted, the corresponding p-value must be lower than the chosen α. The EMO parameters used for solving the test problems are given in Table 7.3.

Table 7.3: EMO parameters used in the test problems discussed in Section 7.3.

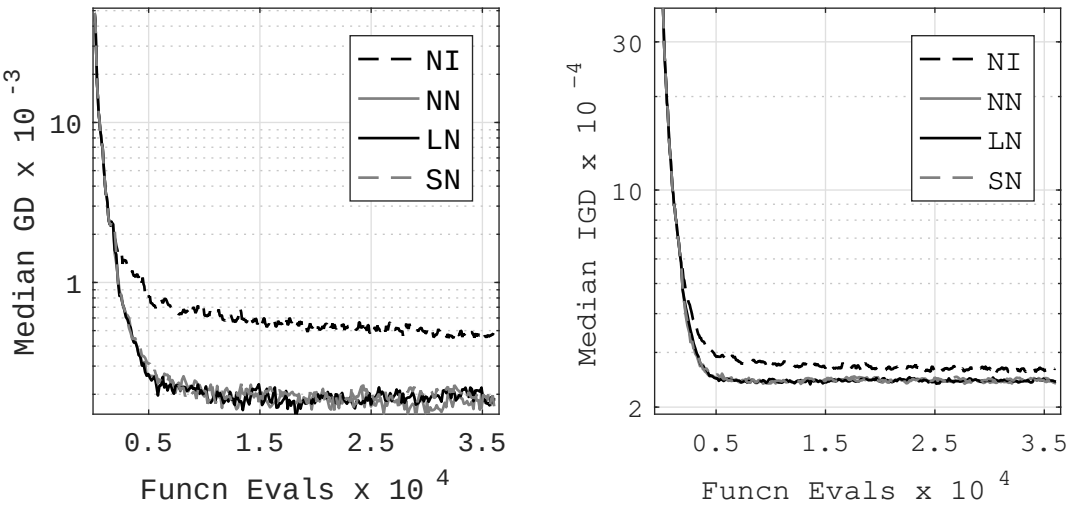
	ZDT1-1	ZDT1-2	ZDT1-3
Population Size	72	52	72
Max Func. Evals	36,000	36,400	50,400
Prob. of Crossover	0.9	0.9	0.9
Prob. of Mutation	1/7	1/5	1/5
Crossover Index	15	15	15
Mutation Index	20	20	20
Start of Learning (% of Max FEs)	5	5	5

7.4.1 ZDT1-1 Results

This problem is designed to test the variable repair strategies NN, LN and SN of Table 7.2 against each other as well as their performance relative to the no-repair case, i.e. NI. Figure 7.3 shows the median GD and IGD results for ZDT1-1 problem. The plots show that EMO/I with any of the three repair strategies NN, LN and SN perform better than the NI strategy in both GD and IGD for same number of objective function evaluations. This observation is supported by the WRS results shown in Table 7.4. The table shows that the three alternate hypothesis namely, $\mathcal{H}_{\text{NN}\prec\text{NI}}$, $\mathcal{H}_{\text{LN}\prec\text{NI}}$ and $\mathcal{H}_{\text{SN}\prec\text{NI}}$, can be accepted at 5% significance level in case of GD as well as IGD in ZDT1-1 problem at the end of maximum function evaluations (36,000 in case of ZDT1-1 problem). Furthermore, Figure 7.3 shows that the variable repair strategies namely, NN, LN and SN, have very similar performance and none can claim to be better than the other in this problem.

Table 7.4: Results of Wilcoxon rank sum test for GD and IGD of ZDT1-1 problem at 36,000 function evaluations and 5% significance level.

		Hypothesis		
		$\mathcal{H}_{\text{NN}\prec\text{NI}}$	$\mathcal{H}_{\text{LN}\prec\text{NI}}$	$\mathcal{H}_{\text{SN}\prec\text{NI}}$
GD	h	Yes	Yes	Yes
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$
IGD	h	Yes	Yes	Yes
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$

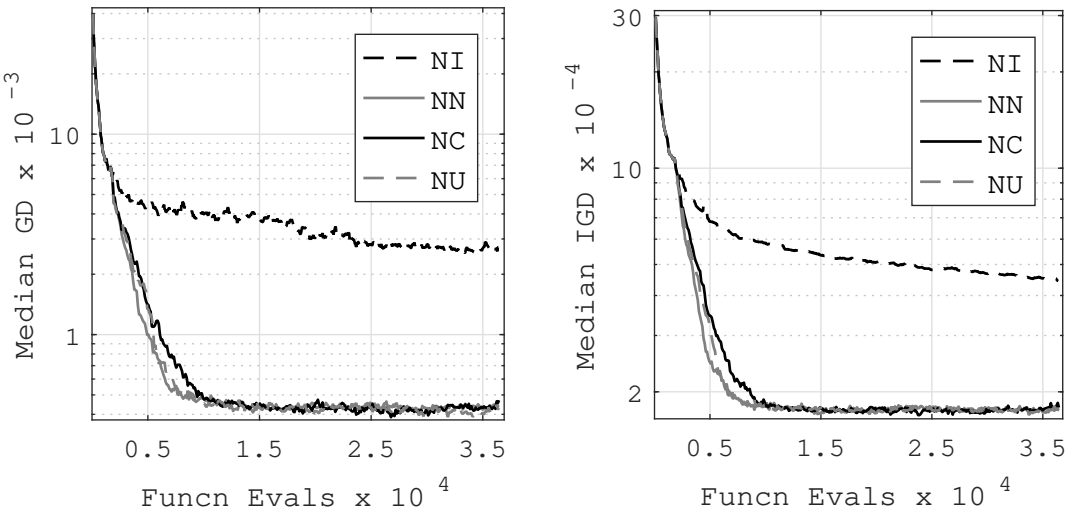


(a) GD results for ZDT1-1. (b) IGD results for ZDT1-1.

Figure 7.3: Median GD and IGD results for ZDT1-1 problem over 30 runs.

7.4.2 ZDT1-2 Results

This problem is designed to test the variable repair strategies NN, NC and NU of Table 7.2 against each other as well as their performance relative to the no-repair case, i.e. NI. Figure 7.4 shows the median GD and IGD results for ZDT1-2 problem. The plots show that EMO/I with any of the three repair strategies NN, NC and NU perform better than the NI strategy in both GD and IGD for same number of objective function evaluations. This observation is supported by the WRS results shown in Table 7.5. The table shows that the three alternate hypothesis namely, $\mathcal{H}_{\text{NN}\prec\text{NI}}$, $\mathcal{H}_{\text{NC}\prec\text{NI}}$ and $\mathcal{H}_{\text{NU}\prec\text{NI}}$, can be accepted at 5% significance level in case of GD as well as IGD in ZDT1-2 problem at the end of maximum function evaluations (36,400 in case of ZDT1-2 problem). Furthermore, Figure 7.4 shows that the variable repair strategies namely, NN, NC and NU, have very similar performance and none can claim to be better than the other in this problem.



(a) GD results for ZDT1-2. (b) IGD results for ZDT1-2.

Figure 7.4: Median GD and IGD results for ZDT1-2 problem over 30 runs.

Table 7.5: Results of Wilcoxon rank sum test for GD and IGD of ZDT1-2 problem at 36,400 function evaluations and 5% significance level.

		Hypothesis		
		$\mathcal{H}_{\text{NN}\prec\text{NI}}$	$\mathcal{H}_{\text{NC}\prec\text{NI}}$	$\mathcal{H}_{\text{NU}\prec\text{NI}}$
GD	h	Yes	Yes	Yes
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$
IGD	h	Yes	Yes	Yes
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$

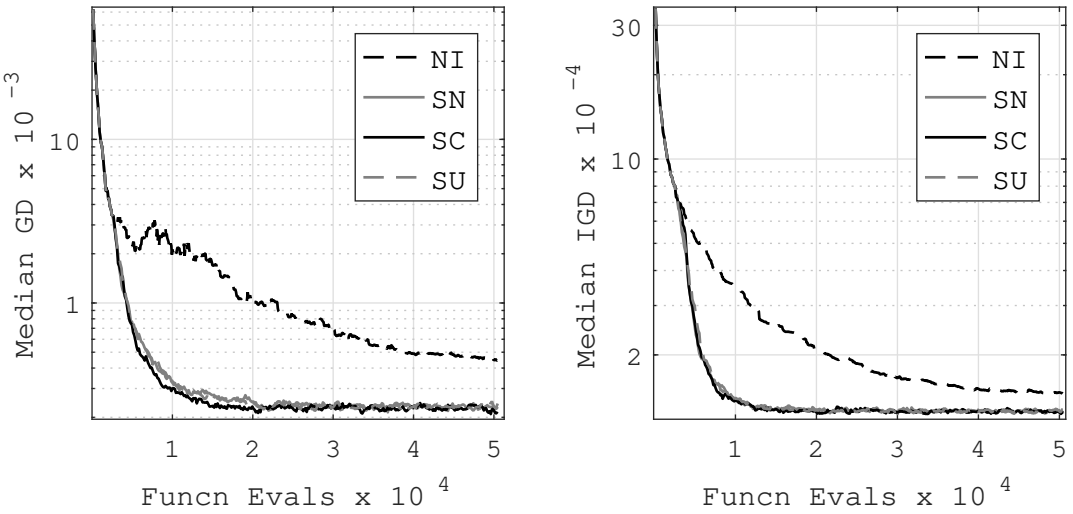
7.4.3 ZDT1-3 Results

This problem is designed to test all nine variable repair strategies of Table 7.2 namely: NN, NC, NU, LN, LC, LU, SN, SC and SU, against each other as well as their performance relative to the no-repair case, i.e. NI. To avoid illegible plots, these results are discussed in three parts.

7.4.3.1 Part-A : Strategies Preferring Short Rules

Figure 7.5 shows the median GD and IGD results for ZDT1-3 problem comparing variable repair strategies SN, SC and SU against each other as well as the no-repair case NI. The

plots show that EMO/I with any of the three repair strategies SN, SC and SU performs better than the NI strategy in both GD and IGD for same number of objective function evaluations. This observation is supported by the WRS test results shown in Table 7.6. The table shows that the three alternate hypothesis namely, $\mathcal{H}_{\text{SN}\prec\text{NI}}$, $\mathcal{H}_{\text{SC}\prec\text{NI}}$ and $\mathcal{H}_{\text{SU}\prec\text{NI}}$, can be accepted at 5% significance level in case of GD as well as IGD in ZDT1-3 problem at the end of corresponding maximum function evaluations (50,400 in case of ZDT1-3 problem). Furthermore, Figure 7.5 shows that the variable repair strategies SN, SC and SU, have very similar performance and none can claim to be better than the other in this problem.



(a) GD results for ZDT1-3.

(b) IGD results for ZDT1-3.

Figure 7.5: Median GD and IGD results for ZDT1-3 problem over 30 runs comparing NI, SN, SC and SU repair strategies of EMO/I.

Table 7.6: Results of Wilcoxon rank sum test for GD and IGD of ZDT1-3 problem comparing EMO/I repair strategies NI, SN, SC and SU at 50,400 function evaluations and 5% significance level.

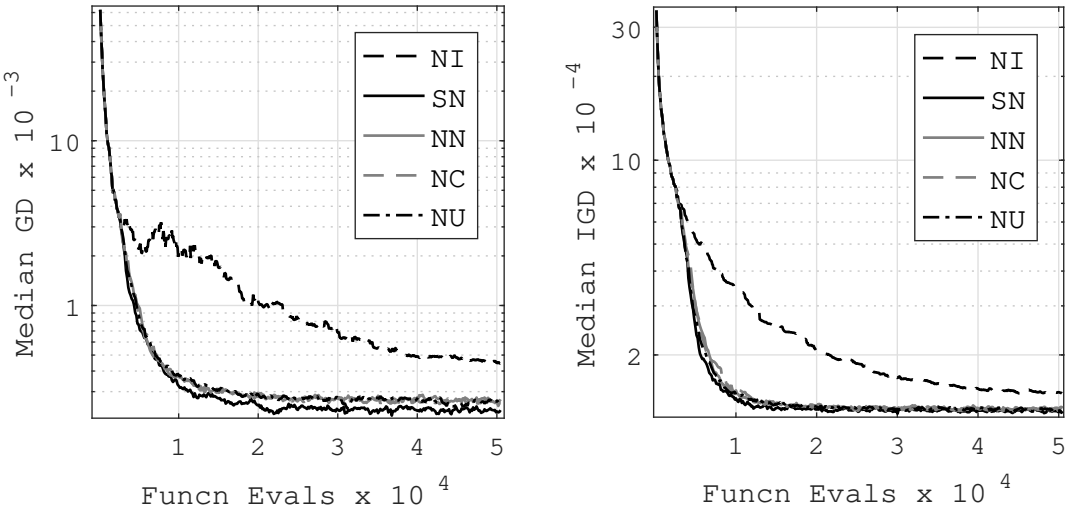
		Hypothesis		
		$\mathcal{H}_{\text{SN}\prec\text{NI}}$	$\mathcal{H}_{\text{SC}\prec\text{NI}}$	$\mathcal{H}_{\text{SU}\prec\text{NI}}$
GD	h	Yes	Yes	Yes
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$
IGD	h	Yes	Yes	Yes
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$

7.4.3.2 Part-B : Strategies with no Preference Based on Length of Rules

Figure 7.6 shows the median GD and IGD results for ZDT1-3 problem comparing variable repair strategies NN, NC, NU and SN against each other as well as the no-repair case NI. The plots show that:

- The repair strategies NN, NC and NU perform better than NI in both GD and IGD. This is backed by results shown in the *Hypothesis (group-I)* column of Table 7.7. Repair strategy SN too performs better than NI but the corresponding results are already shown in Section 7.4.3.1.

- Furthermore, the repair strategy SN performs better than NN, NC and NU in terms of GD whereas their performance in terms of IGD metric are similar. The corresponding WRS test results are shown in *Hypothesis (group-II)* column of Table 7.7.



(a) GD results for ZDT1-3. (b) IGD results for ZDT1-3.

Figure 7.6: Median GD and IGD results for ZDT1-3 problem over 30 runs comparing NI, SN, NN, NC and NU repair strategies of EMO/I.

Table 7.7: Results of Wilcoxon rank sum test for GD and IGD of ZDT1-3 problem comparing EMO/I repair strategies NN, NC, NU, SN and NI at 50,400 function evaluations and 5% significance level.

		Hypothesis (group-I)			Hypothesis (group-II)		
		$\mathcal{H}_{\text{NN}\prec\text{NI}}$	$\mathcal{H}_{\text{NC}\prec\text{NI}}$	$\mathcal{H}_{\text{NU}\prec\text{NI}}$	$\mathcal{H}_{\text{SN}\prec\text{NN}}$	$\mathcal{H}_{\text{SN}\prec\text{NC}}$	$\mathcal{H}_{\text{SN}\prec\text{NU}}$
GD	h	Yes	Yes	Yes	Yes	Yes	Yes
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	2.0×10^{-3}	3.6×10^{-5}	1.1×10^{-5}
IGD	h	Yes	Yes	Yes	Inconclusive		
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$			

7.4.3.3 Part-C : Strategies Preferring Long Rules

Figure 7.7 shows the median GD and IGD results for ZDT1-3 problem comparing variable repair strategies LN, LC, LU and SN against each other as well as the no-repair case NI. The plots show that:

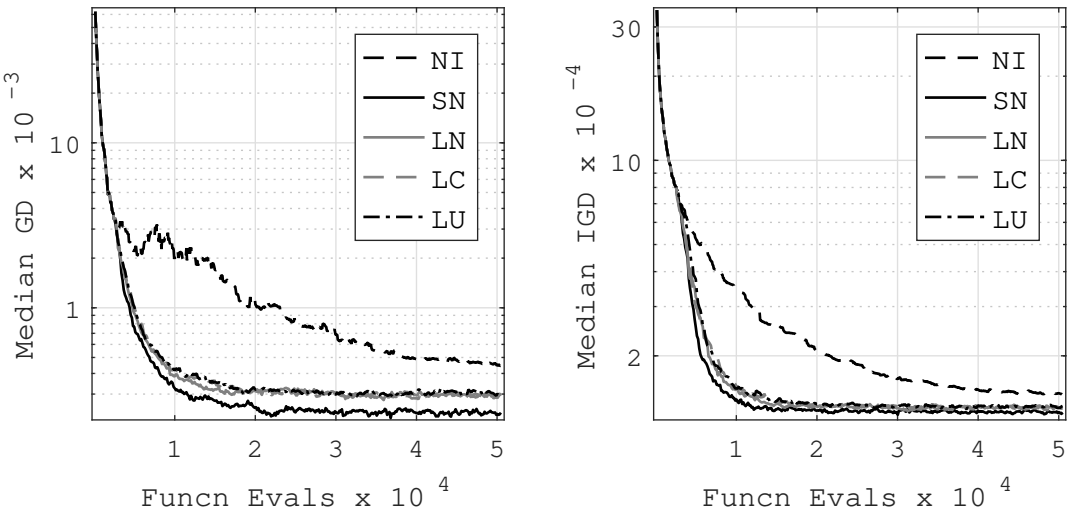
- The repair strategies LN, LC and LU perform better than NI in both GD and IGD. This is backed by results shown in the *Hypothesis (group-I)* column of Table 7.8. Repair strategy SN too performs better than NI but the corresponding results are already shown in Section 7.4.3.1.

- Furthermore, the repair strategy SN performs better than LN, LC and LU in terms of GD whereas their performance in terms of IGD metric are similar. The corresponding WRS test results are shown in *Hypothesis (group-II)* column of Table 7.8.

7.4.4 Summary of Results on Test Problems

Based on the results shown in Sections 7.4.1, 7.4.2 and 7.4.3, we can conclude the following:

- EMO/I with any of the nine variable repair strategies, namely, NN, NU, NC, SN, SU, SC, LN, LU and LC, performs better than no variable repair case (NI), in terms of GD



(a) GD results for ZDT1-3. (b) IGD results for ZDT1-3.

Figure 7.7: Median GD and IGD results for ZDT1-3 problem over 30 runs comparing LN, LC, LU, SN and NI repair strategies of EMO/I.

Table 7.8: Results of Wilcoxon rank sum test for GD and IGD of ZDT1-3 problem comparing EMO/I repair strategies LN, LC, LU, SN and NI at 50,400 function evaluations and 5% significance level.

		Hypothesis (group-I)			Hypothesis (group-II)		
		$\mathcal{H}_{\text{LN}\prec\text{NI}}$	$\mathcal{H}_{\text{LC}\prec\text{NI}}$	$\mathcal{H}_{\text{LU}\prec\text{NI}}$	$\mathcal{H}_{\text{SN}\prec\text{LN}}$	$\mathcal{H}_{\text{SN}\prec\text{LC}}$	$\mathcal{H}_{\text{SN}\prec\text{LU}}$
GD	h	Yes	Yes	Yes	Yes	Yes	Yes
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$
IGD	h	Yes	Yes	Yes	Inconclusive		
	p	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$			

and IGD on all three test problems.

- Results of Section 7.4.3 show that EMO/I with the variable repair strategy SN performs better than the repair strategies NN, NU, NC, LN, LU and LC in terms of GD. In terms of IGD, there is no single clear winner.

Following the implication of above summary, we compare the performance of EMO/I with SN repair strategy with that of no-repair NI strategy on three engineering design MOO problems presented in following sections.

7.5 Engineering Problems

This section describes three MOO problems from engineering design. All three problems are slightly modified from their original form so that there are no transition points in the final PO front. Doing so makes all the rules among PO solutions set to be applicable for the entire PO front.

7.5.1 Two Bar Truss Problem (TBT)

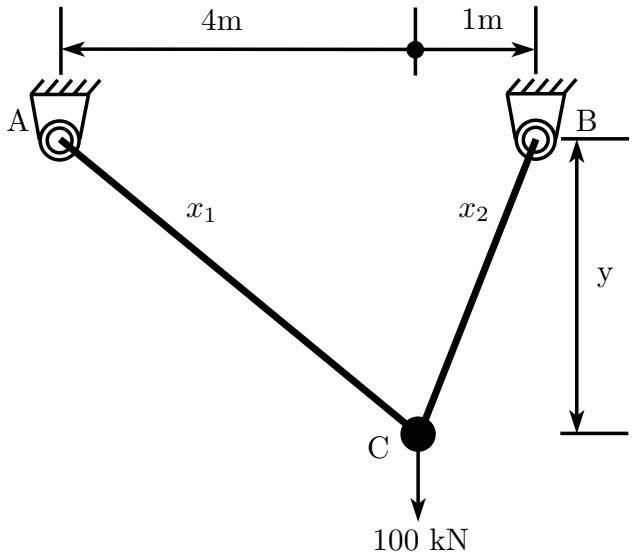


Figure 7.8: A two membered truss structure.

In this problem, a two bar truss structure shown in Figure 7.8 is designed to carry a certain load without elastic failure. The truss members have cross sectional area x_1 m^2 and x_2 m^2 respectively and the point of loading is at a vertical distance of x_3 m from the truss member hinge. The two objectives, both to be minimized, are the volume of the structure and the maximum stress developed in its two members. Its mathematical formulation is given by Equation (7.23). The original formulation of the problem given in [31] did not contain the constraint g_2 given in Equation (7.23). The constraint g_2 is added to avoid any

transition points in the PO front so that rules found are applicable to all the PO solutions. The transition point in this problem is encountered because the cross sectional area x_2 hits its upper limit in order to minimize both objectives in a PO way. The reader is referred to [10] for more details on the reason and location of transition point at $V \approx 0.044\, m^3$.

$$\begin{aligned}
\text{Minimize} \quad & V = x_1\sqrt{16 + x_3^2} + x_2\sqrt{1 + x_3^2}, \\
& S = \max(\sigma_{AC}, \sigma_{BC}), \\
\text{Subject to} \quad & g_1 \equiv \max(\sigma_{AC}, \sigma_{BC}) \leq 10^5, \\
& g_2 \equiv V \leq 0.044, \\
\text{Where} \quad & \sigma_{AC} = \frac{20\sqrt{16 + x_3^2}}{x_3 x_1}, \sigma_{BC} = \frac{80\sqrt{1 + x_3^2}}{x_3 x_2}, \\
& x_1, x_2 \in [0, 0.01], \quad x_3 \in [1.0, 3.0].
\end{aligned} \tag{7.23}$$

Figure 7.9 shows the PO front for this problem obtained using NSGA-II. To ascertain that the black curve shown in Figure 7.9 is indeed the true PO front, a local search is run from six points (shown with gray circles) using ϵ-constraint method [109]. An ϵ-constraint is set on the volume objective while maximum stress objective is minimized with local search. As can be seen, the local search is not able find any point very different from the start point. Furthermore, the new front obtained by running a local search from all the points of NSGA-II front amounts to a minor increase of 0.0132% in the Hypervolume [31]. Thus, the front of Figure 7.9 can safely be assumed to be very close to the idea PO front and hence it is used for evaluating GD and IGD value results in later sections.

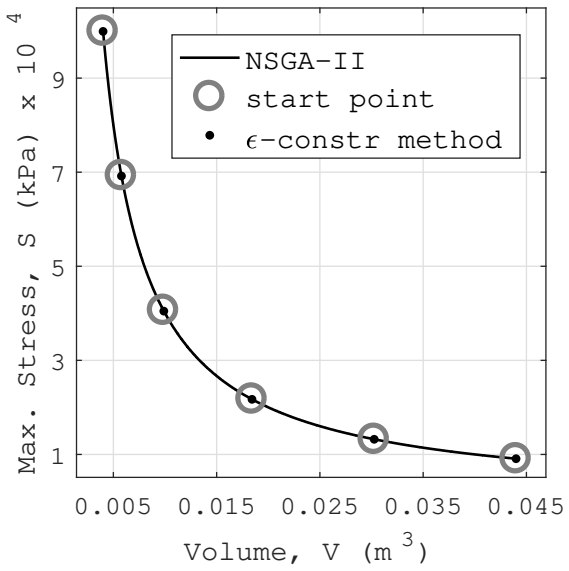


Figure 7.9: Pareto-optimal front for the two member truss problem.

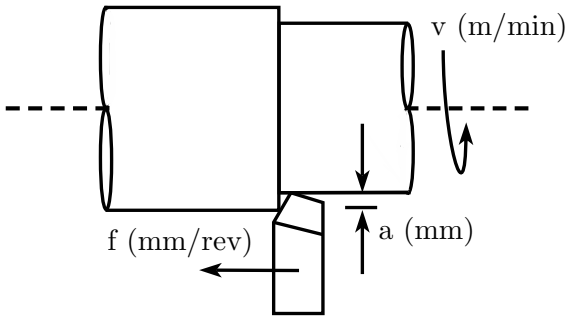


Figure 7.10: A diagram of metal turning process.

7.5.2 Metal Cutting Problem (MC)

This MOO problem involves optimizing the twin objectives of operation time (T_p) and tool life (ξ) during the turning process of a steel bar on a CNC lathe with a P20 carbide tool as shown in Figure 7.10. Equation (7.24) presents the mathematical formulation of this problem. The formulation given in Equation (7.24) contains an additional constraint g_4 compared to the original formulation of the problem given in [110]. This constraint on the operation time objective is put to avoid transition points in the PO front so that the rules found are applicable to all PO solutions. The location of this transition point at $T_p \approx 0.9545$

minutes is discovered manually in [29].

$$
\begin{aligned}
\text{Minimize} \quad & T_p(\mathbf{x}), \\
& \xi(\mathbf{x}), \\
\text{Subject to} \quad & g_1(\mathbf{x}) \equiv 1 - \frac{P(\mathbf{x})}{\eta P^{max}} \geq 0, \quad g_2(\mathbf{x}) \equiv 1 - \frac{F_c(\mathbf{x})}{F_c^{max}} \geq 0, \\
& g_3(\mathbf{x}) \equiv 1 - \frac{R(\mathbf{x})}{R^{max}} \geq 0, \quad g_4(\mathbf{x}) \equiv T_p(\mathbf{x}) \leq 0.9545, \\
\text{Where} \quad & T_p(\mathbf{x}) = 0.15 + 219{,}912 \left(\frac{1 + \frac{0.20}{T(\mathbf{x})}}{MRR(\mathbf{x})} \right) + 0.05, \quad \xi(\mathbf{x}) = \frac{21{,}991{,}200}{MRR(\mathbf{x}) T(\mathbf{x})}, \\
& T(\mathbf{x}) = \frac{5.48 \times 10^9}{v^{3.46} f^{0.696} a^{0.46}}, \quad F_c(\mathbf{x}) = \frac{6.56 \times 10^3 f^{0.917} a^{1.10}}{v^{0.286}}, \\
& P(\mathbf{x}) = \frac{v F_c}{60{,}000}, \quad MRR(\mathbf{x}) = 1000\, vfa, \quad R(\mathbf{x}) = \frac{125 f^2}{r_n}, \\
& P^{max} = 10 \text{ kW}, \; F_c^{max} = 5000 \text{ N}, \; R^{max} = 50 \; \mu\text{m}, \\
& r_n = 0.8 \text{ mm}, \; \eta = 0.75, \\
& v \equiv x_1 \in [250.0, 400.0] \text{ in m/min}, \\
& f \equiv x_2 \in [0.15, 0.55] \text{ in mm/rev and}, \\
& a \equiv x_3 \in [0.5, 6.0] \text{ in mm}.
\end{aligned}
\tag{7.24}
$$

Figure 7.11 shows the PO front for this problem obtained using NSGA-II. To ascertain that the black curve shown in Figure 7.11 is indeed the true PO front, a local search is run from six points (shown with gray circles) using ϵ-constraint method. An ϵ-constraint is set on the operation time objective, T_p, while the used tool life objective, ξ, is minimized with local search. As can be seen, the local search is not able find any point very different from the start point. Furthermore, the new front obtained by running a local search from all the points of NSGA-II front amounts to a minor increase of 0.0692 % in the Hypervolume. Thus

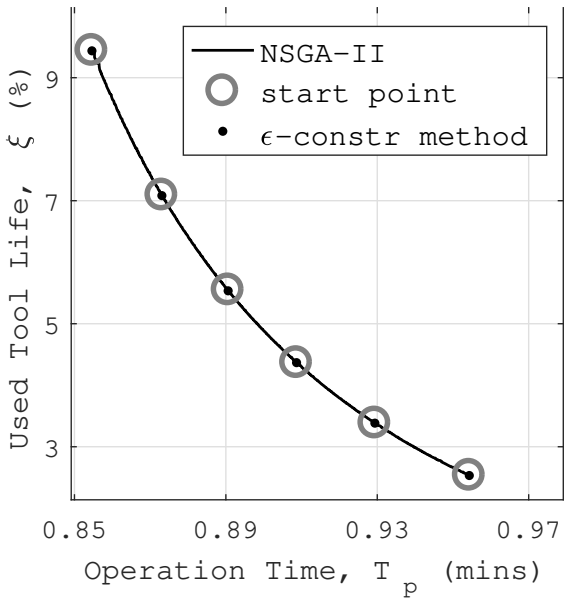


Figure 7.11: Pareto-optimal front for the Metal Cutting problem.

the front of Figure 7.11 can safely be assumed to be very close to the idea PO front and hence it is used for evaluating GD and IGD value results in later sections.

7.5.3 Welded Beam Problem (WB)

In this problem, a beam needs to be welded to another beam and must carry a load F [10]. It is desired to find four design parameters (thickness of the beam, b, width of the beam t, length of weld l, and weld thickness h) for which both, the cost of the beam C and the vertical deflection at the end of the beam D, is minimized. Its mathematical formulation is given by Equation (7.25). The original formulation of the problem did not contain the constraint g_5 given in Equation (7.25). The constraint g_5 is added to avoid any transition points in the PO front so that rules found are applicable to all the PO solutions. The transition point in this problem is encountered because beyond a deflection of 0.004 inches, weld thickness h encounters its maximum possible value, i.e. beam thickness b. This value is obtained using

manual innovization as outlined in [10].

$$
\begin{aligned}
\text{Minimize} \quad & C(\mathbf{x}) = 1.10471\, h^2 l + 0.04811\, tb(14.0 + l), \\
& D(\mathbf{x}) = \frac{2.1952}{t^3 b}, \\
\text{Subject to} \quad & g_1(\mathbf{x}) \equiv 13{,}600 - \tau(\mathbf{x}) \geq 0, \quad g_2(\mathbf{x}) \equiv 30{,}000 - \sigma(\mathbf{x}) \geq 0, \\
& g_3(\mathbf{x}) \equiv b - h \geq 0, \quad g_4(\mathbf{x}) \equiv P_c(\mathbf{x}) - 6000 \geq 0, \\
& g_5(\mathbf{x}) \equiv 0.004 - D(\mathbf{x}) \geq 0, \\
\text{Where} \quad & \tau(\mathbf{x}) = \sqrt{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')/\sqrt{(0.25(l^2 + (h+t)^2))}}, \\
& \tau' = 6000/\sqrt{2}hl, \\
& \tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)^2}}{2\{0.707hl(l^2/12 + 0.25(h+t)^2)\}}, \\
& \sigma(\mathbf{x}) = 504{,}000/t^2 b, \\
& P_c(\mathbf{x}) = 64{,}746.022(1 - 0.0282346t)tb^3, \\
& 0.125 \leq h, b \leq 5.0 \text{ inches}, \quad 0.1 \leq l, t \leq 10.0 \text{ inches}.
\end{aligned}
\tag{7.25}
$$

Figure 7.12 shows the PO front for this problem obtained using NSGA-II. To ascertain that the black curve shown in Figure 7.12 is indeed the true PO front, a local search is run from six points (shown with gray circles) using ϵ-constraint method. An ϵ-constraint is set on the cost objective, C, while the deflection objective, D, is minimized with local search. As can be seen, the local search is not able find any point very different from the start points. Furthermore, the new front obtained by running a local search from all the points of NSGA-II front amounts to a minor increase of 0.0017 % in the Hypervolume. Thus the front of Figure 7.12 can safely be assumed to be very close to the idea PO front and hence

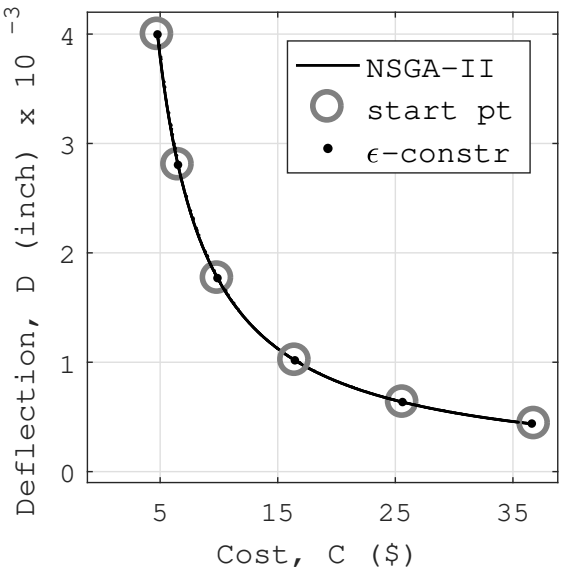


Figure 7.12: Pareto optimal front for the WB problem.

it is used for evaluating GD and IGD results in later sections.

The EMO parameters used for these engineering design problems are shown in Table 7.9. Next we discuss the performance of no repair strategy NI and variable repair strategy SN from Table 7.2 in solving these engineering design problems.

Table 7.9: EMO parameters used for solving the engineering problems discussed in Section 7.5.

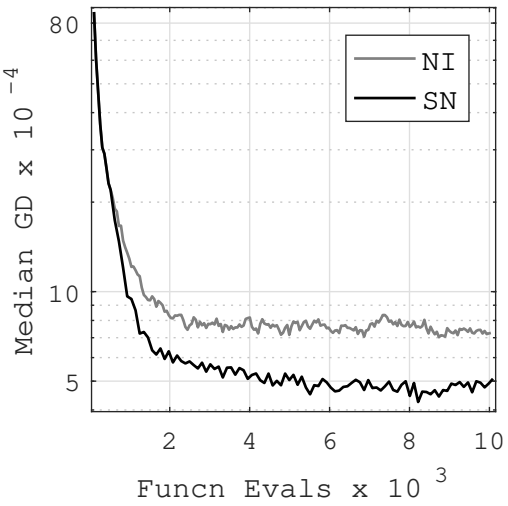
Parameter Name	TBT	MC	WB
Population Size	52	72	92
Max Func. Evals	10,036	36,000	25,024
Prob. of Crossover	0.9	0.9	0.9
Prob. of Mutation	1/3	1/3	1/4
Crossover Index	10	10	10
Mutation Index	50	50	50
Start of Learning (% of Max FEs)	5	5	5

7.6 Results on Engineering Problems

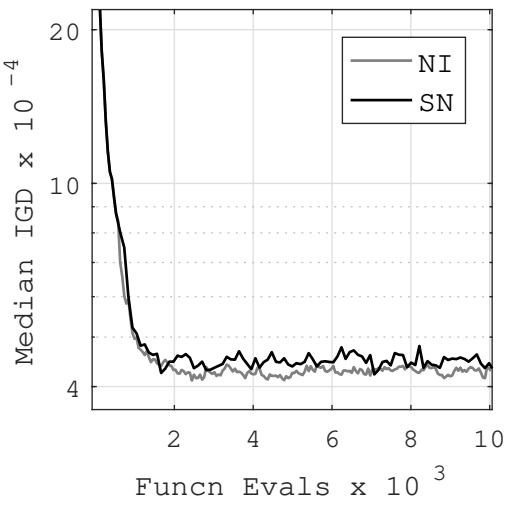
The repair strategy SN was shown to be the best performer among all other repair strategies (in Table 7.2) on test problems at the end of Section 7.4. Hence, this section compares the performance of an EMO algorithm with that of an EMO/I algorithm with repair strategy SN. As in Section 7.4, the different algorithms are compared on the metrics of median GD and median IGD over 30 runs. All claims of one algorithm being better than the other are backed with results of WRS test of statistical significance. As before, $\mathcal{H}_{\mathrm{A}\prec\mathrm{B}}$ represents the left tailed hypothesis test, where the alternative hypothesis states that the median of distribution A is lower than the median of distribution B at some significance level α. A significance level of $\alpha = 5\%$ is used in all the statistical tests. For all the statistical tests, both h and p values are shown in the results. An h-value of No means that the aforementioned alternative hypothesis cannot be accepted at the desired significance level and an h-value of Yes means otherwise. Also, for the aforementioned alternative hypothesis to be accepted, the corresponding p-value must be lower than the chosen α.

7.6.1 Two Bar Truss Problem Results

Figure 7.13 shows the median GD and IGD results for TBT problem. The plots show that EMO/I with SN repair strategy performs better than the NI strategy in terms of GD. This observation is supported by the WRS test results shown in the TBT column of Table 7.10. In terms of IGD, the NI strategy performs marginally better. This is expected as EMO/I is designed to focus more on convergence than diversity. Figure 7.14 shows PO front for the best GD case out of the thirty runs for EMO/I with SN strategy in TBT problem overlaid with PO front of TBT problem taken from Figure 7.9.



(a) GD results for TBT problem.



(b) IGD results for TBT problem.

Figure 7.13: Median GD and IGD results for TBT problem over 30 runs.

Table 7.10: Results of Wilcoxon rank sum test for GD of TBT, MC and WB problem at their respective maximum function evaluations and 5% significance level.

		Hypothesis		
		TBT	MC	WB
		$\mathcal{H}_{\text{SN}\prec\text{NI}}$	$\mathcal{H}_{\text{SN}\prec\text{NI}}$	$\mathcal{H}_{\text{SN}\prec\text{NI}}$
GD	h	Yes	Yes	Yes
	p	$< 10^{-6}$	1.2×10^{-4}	$< 10^{-6}$

7.6.1.1 Rules Found in TBT Problem

The rules detected and used in repairing the PO solutions of TBT problem towards the end of EMO/I (with SN) in the best GD run are:

$$x_1^{-1.0082} \cdot x_2 = 2.075 \text{ and } x_3 = 1.9449 \pm 0.0674. \tag{7.26}$$

Figures 7.15 and 7.16 show the PO solutions (corresponding to the PO front shown in Figure 7.9) shown with black points, overlaid with the learned variable relations of Equation (7.26) shown with a gray line. The two overlaid plots are very close, thus a close up of a part of the plot is shown in the inset for clarity. The two learned rules of Equation (7.26)

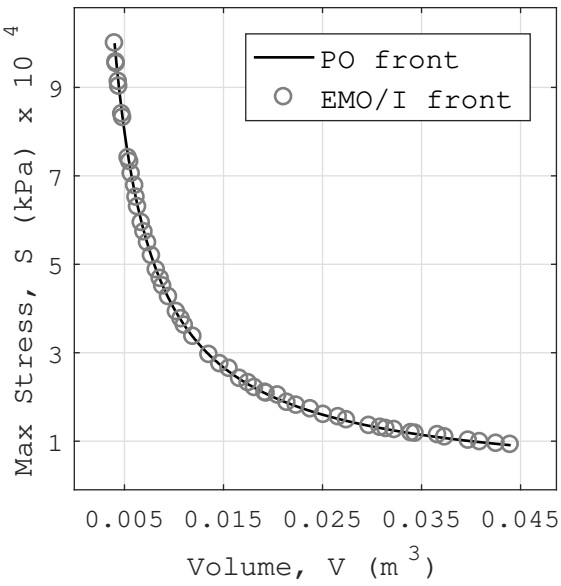


Figure 7.14: Front coverage by EMO/I method in TBT problem for the best GD run.

are very close to the analytical relations that exist in the PO solutions of TBT problem, i.e. $x_2/x_1 = 2.0$ and $x_3 = 2.0$ [10].

7.6.2 Metal Cutting Problem Results

Figure 7.17 shows the median GD and IGD results for MC problem. The plots show that EMO/I with SN repair strategy performs better than the NI strategy in terms of GD. This observation is supported by the WRS test results shown in the MC column of Table 7.10. In terms of IGD, the NI strategy performs marginally better. This is confirmed in Figure 7.18 that shows PO front for the best GD case out of the thirty runs for EMO/I (with SN strategy) in MC problem overlaid with PO front of MC problem taken from Figure 7.11. This is expected as EMO/I is designed to focus more on convergence than diversity.

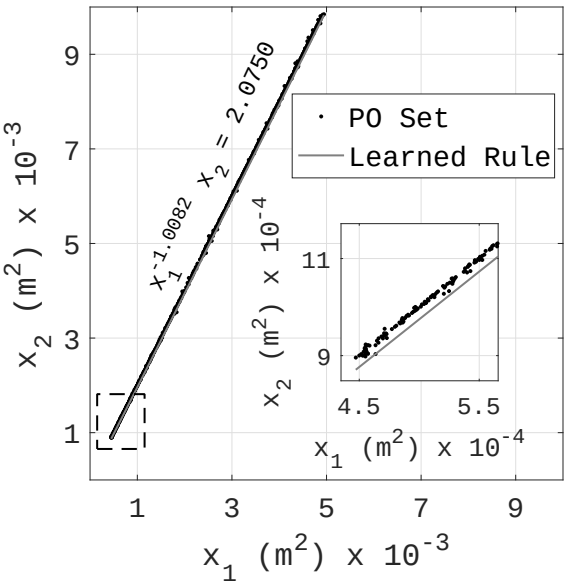


Figure 7.15: Rule between variables x_1 and x_2 identified by EMO/I in TBT problem at the end of best GD run.

7.6.2.1 Rules Found in MC Problem

The rules detected and used in repairing the PO solutions of MC problem towards the end of EMO/I (with SN) in the best GD run are:

$$f = 0.55 \pm 5.4 \times 10^{-6} \text{ and } v \cdot a^{1.5420} = 804.6058. \tag{7.27}$$

Figures 7.19 and 7.20 show the PO solutions (corresponding to the PO front shown in Figure 7.11) shown with black points, overlaid with the learned variable relations of Equation (7.27) shown with a gray line. The two overlaid plots are very close, thus a close up of a part of the plot is shown in the inset for clarity.

7.6.3 Weld Beam Problem Results

Figure 7.21 shows the median GD and IGD results for MC problem. The plots show that EMO/I with SN repair strategy performs better than the NI strategy in terms of GD. This

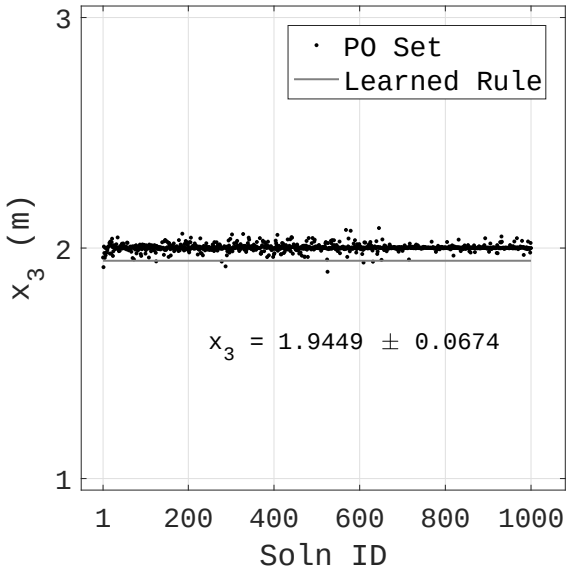


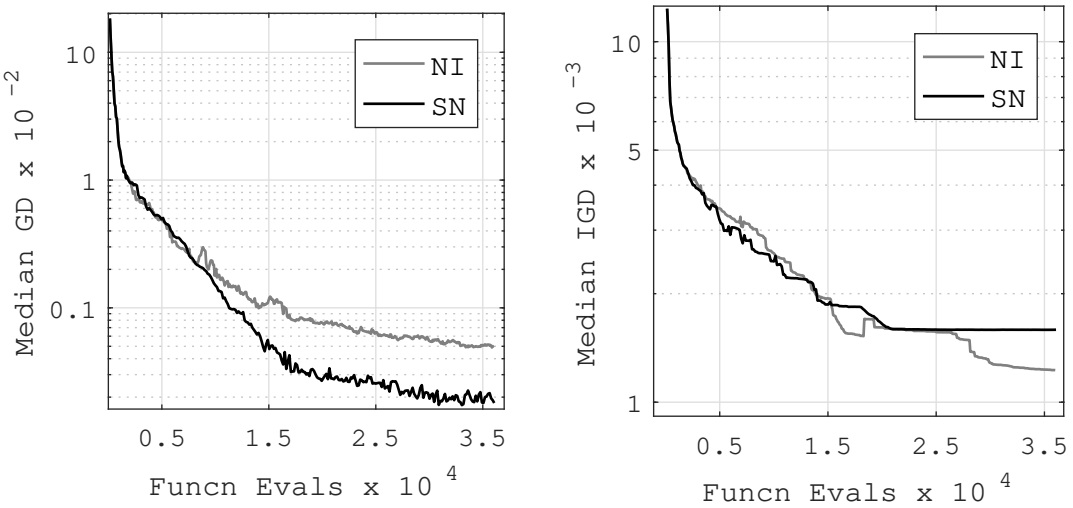
Figure 7.16: Rule for variable x_3 identified by EMO/I in TBT problem at the end of best GD run.

observation is supported by the WRS test results shown in the WB column of Table 7.10. In terms of IGD, the SN strategy appears to perform better (Figure 7.21b) but NI catches up towards the end. Also, for the intermediate function evaluations where SN appears to have a lower IGD is not supported by WRS test results. Figure 7.22 that shows PO front for the best GD case out of the thirty runs for EMO/I (with SN strategy) in WB problem overlaid with PO front of WB problem taken from Figure 7.12.

7.6.3.1 Rules Found in WB Problem

The rules detected and used in repairing the PO solutions of WB problem towards the end of EMO/I (with SN) in the best GD run are:

$$h \cdot l^{0.8842} = 0.9386 \ , \ h \cdot b^{-0.5600} = 0.7239 \ , \ l \cdot b^{-0.6323} = 1.3408 \ , \ t = 9.9939 \pm 0.0087. \tag{7.28}$$



(a) GD results for MC problem.

(b) IGD results for MC problem.

Figure 7.17: Median GD and IGD results for MC problem over 30 runs.

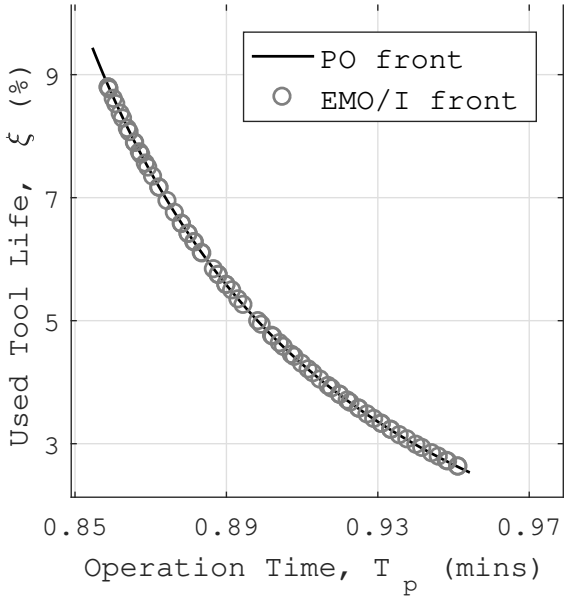


Figure 7.18: Front coverage by EMO/I method in MC problem for the best GD run.

Figures 7.23, 7.24, 7.25 and 7.26 show the PO solutions (corresponding to the PO front shown in Figure 7.12) shown with black points, overlaid with the learned variable relations of Equation (7.28) shown with a gray line. The two overlaid plots are very close, thus a close up of a part of the plot is shown in the inset for clarity.

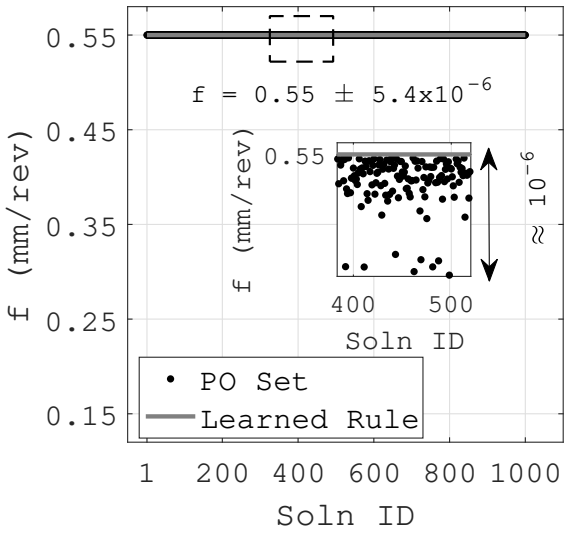


Figure 7.19: Rule for variable f identified by EMO/I in MC problem at the end of best GD run.

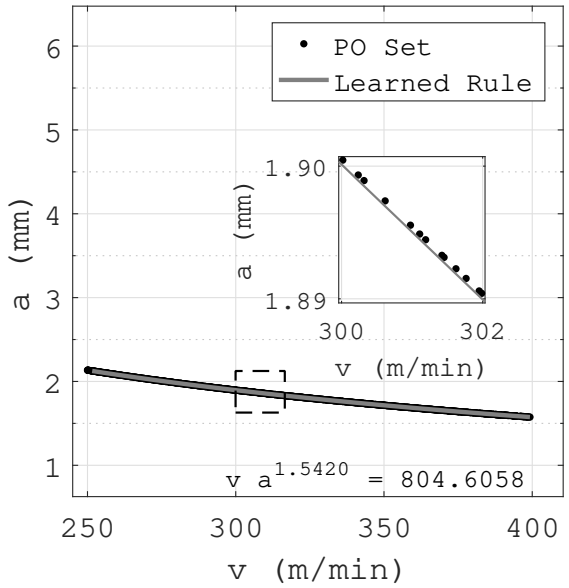
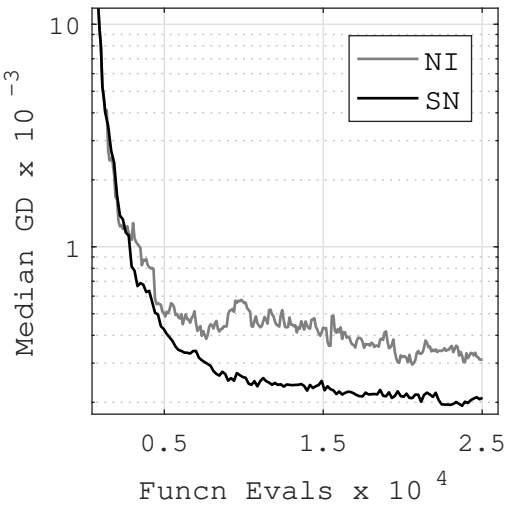


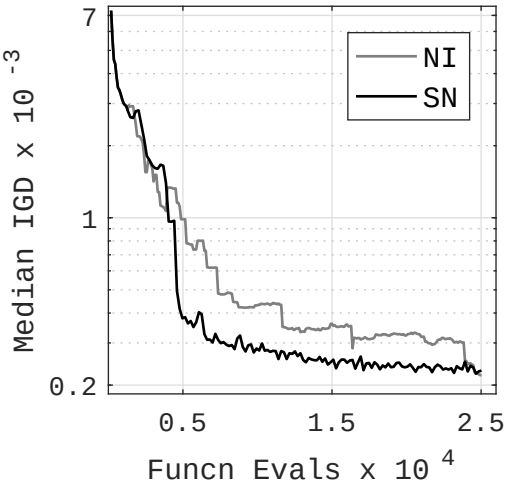
Figure 7.20: Rule among variables v and a identified by EMO/I in MC problem at the end of best GD run.

7.7 Concluding Remarks

The results in Sections 7.4 and 7.6 show that the idea of online innovization can be useful in expediting convergence of EMO algorithms, however this may come at an expense of the diversity achieved in final PO front. The PO fronts of all the problems that we have



(a) GD results for WB problem.



(b) IGD results for WB problem.

Figure 7.21: Median GD and IGD results for WB problem over 30 runs.

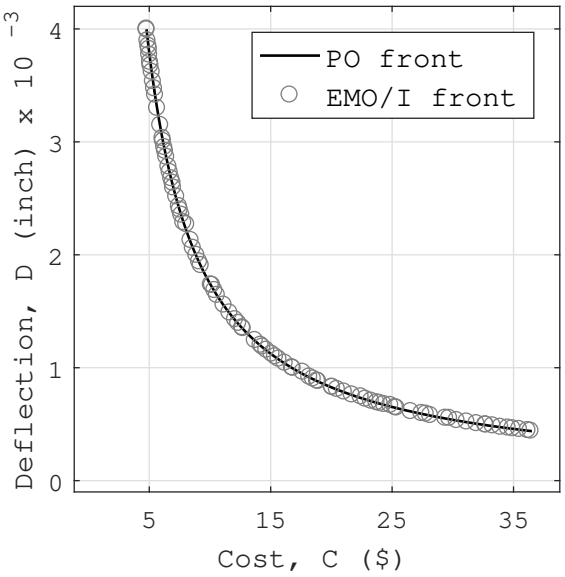


Figure 7.22: Front coverage by EMO/I method in WB problem for the best GD run.

considered till now do not have any transition points. Also, the rules on which we tested the EMO/I idea were composed solely of MOO problem's variables and no rules were contained any other possible basis functions in them such as MOO problem's objective values or constraints. In the coming chapter, we will look at ideas that can address these issues.

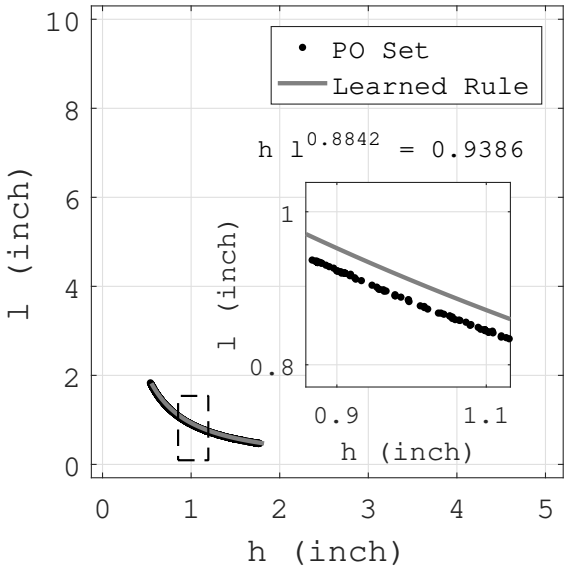


Figure 7.23: Rule between variables h and l identified by EMO/I in WB problem at the end of best GD run.

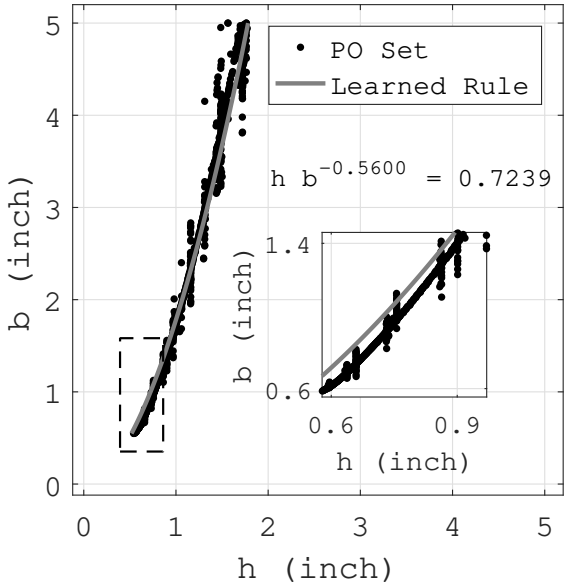


Figure 7.24: Rule between variables h and b identified by EMO/I in WB problem at the end of best GD run.

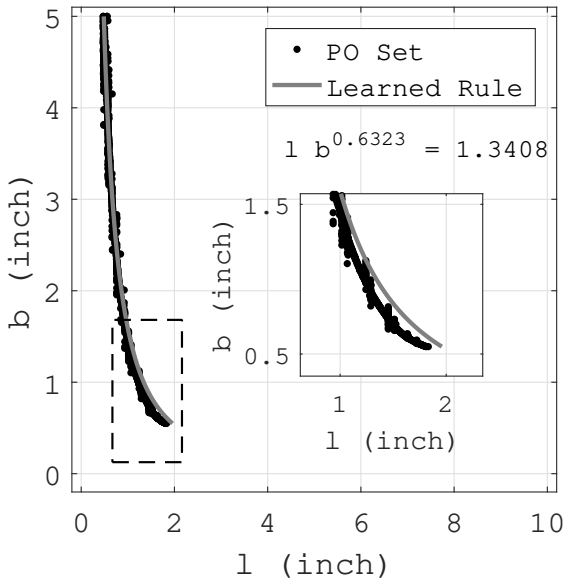


Figure 7.25: Rule between variables l and b identified by EMO/I in WB problem at the end of best GD run.

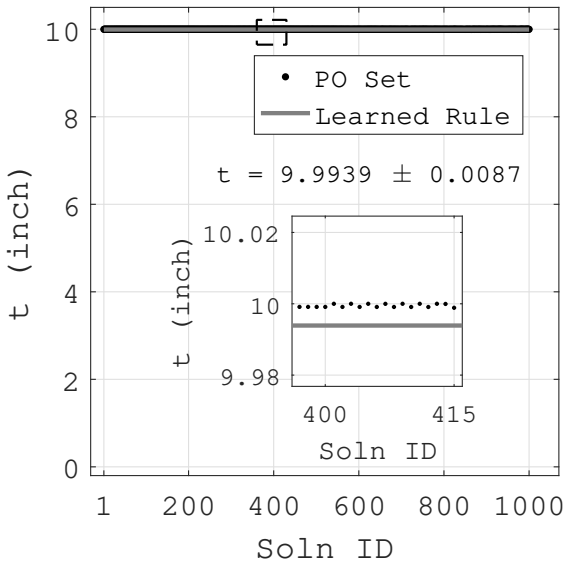


Figure 7.26: Rule for variable t identified by EMO/I in WB problem at the end of best GD run.

Chapter 8

Issues of Transition Points and Repairs Based on Complex Rules

8.1 Handling Transition Points

Until now, we have avoided dealing with transition points in the PO fronts and assumed that whatever rules we find are applicable to all the solutions of a PO front. However, this is not the case in practice and usually PO solutions to MOO problems contain one or more *transition points* [10]. A transition point is a point on a PO front across which the design rules being adhered to by the solutions, change significantly. Although such "points" can be lines or regions in case the PO front is a surface in three or more dimensional objective space, in case of bi-objective problems, these invariably are points and hence we will stick with the term transition point or TP from hereon. In problems with transition points present in the PO front, our methods developed in Chapter 7 may not work because there we assumed that the rules being learned are applicable to the entire front. In this section, we will see how can we learn rules and repair solutions as part of EMO/I in the presence of transition points. Let us first try to understand the concept of a transition point with an example.

Equation (8.1) shows another version of the two bar truss problem. Recall that this truss problem is also presented earlier in Section 7.5.1 by Equation (7.23). Note that the problem

description given in Equation (8.1) has one less constraint, namely $g_2 \equiv V \leq 0.044$, than the problem given in Equation (7.23).

$$\begin{aligned}
\text{Minimize} \quad & V = x_1\sqrt{16 + x_3^2} + x_2\sqrt{1 + x_3^2}, \\
& S = \max(\sigma_{AC}, \sigma_{BC}), \\
\text{Subject to} \quad & g_1 \equiv \max(\sigma_{AC}, \sigma_{BC}) \leq 10^5, \\
\text{Where} \quad & \sigma_{AC} = \frac{20\sqrt{16 + x_3^2}}{x_3 x_1}, \sigma_{BC} = \frac{80\sqrt{1 + x_3^2}}{x_3 x_2}, \\
& x_1, x_2 \in [0, 0.01], \quad x_3 \in [1.0, 3.0].
\end{aligned} \tag{8.1}$$

This change in problem description introduces a transition point in the PO front of the problem. Figure 8.1 shows this transition point in the objective space and Figure 8.2 shows it in the variable space.

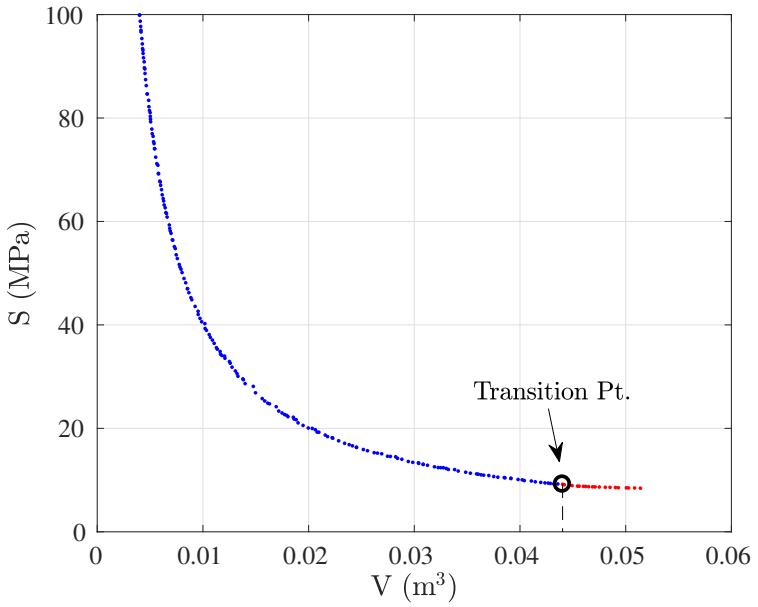


Figure 8.1: Transition point encountered in Truss problem shown in objective space.

As is analytically and numerically shown in [10], the location of this transition point in the objective space is ($V = 0.04472$ m^3, $S = 8.94426$ MPa) and in the variable space is

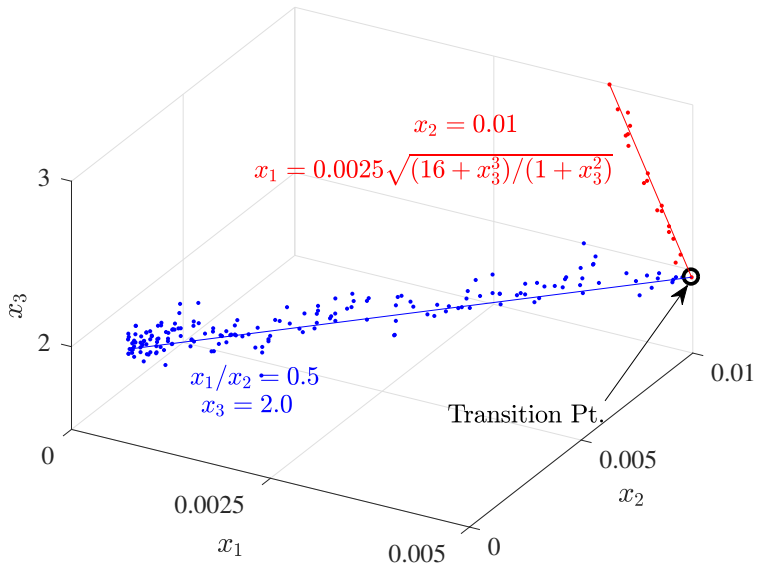


Figure 8.2: Transition point encountered in Truss problem shown in variable space.

($x_1 = 0.005$ m, $x_2 = 0.01$ m, $x_3 = 2.0$ m). The reason for such a transition point to appear is as follows. Until variable x_2 reaches its upper bound of 0.01m, to achieve a solution with smaller maximum stress, S (and larger volume V) optimally, both cross sections x_1 and x_2 need to be increased linearly in the ratio of $x_1/x_2 = 0.5$ while keeping $x_3 = 2.0$. Hence Equation (8.2) shows the rules that exist before encountering the transition point. This region is shown in blue in Figures 8.1 and 8.2.

$$x_1/x_2 = 0.5,\ y = 2.0\text{m} \tag{8.2}$$

Beyond this critical point (T), since x_2 cannot be increased any further, the only way to reduce the stresses is to increase the length x_3 in a manner so as to make the stresses in both members equal. An increase of x_3 increases the length of the members, but decreases the component of the applied load on each member. Thus, a smaller cross-sectional area can be used to withstand the smaller load causing a smaller developed stress. Equation (8.3) shows the rules that the variables adhere to in order to reduce the maximum stress (while

increasing volume) in an optimal manner. This region is shown in red in Figures 8.1 and 8.2.

$$x_2 = 0.01\text{m},\ x_1 = 0.0025\sqrt{\frac{16 + x_3^2}{1 + x_3^2}}\ \text{m} \tag{8.3}$$

Now, if we attempt to apply the method of rule learning and variable repair discussed in Chapter 7 directly to all non-dominated solutions of a MOO problem using EMO/I, we will learn the wrong rule parameters, because we will be trying to learn a rule using data of all non-dominated solutions instead of over individual partitions. As seen in case of two bar truss problem, the rules are very different across the transition point shown in Figure 8.1 and 8.2. Furthermore, we will also make poor repairs, waste function evaluations and worsen the algorithm performance. To avoid this, the following is necessary:

- Detect disjoint *regions* of PO solutions separated by transition points,
- Learn rules separately for each of these regions, and
- Repair solutions belonging to different regions using learned rules of corresponding regions. If no rules of desired quality can be learned from a region, then do not attempt to repair solutions of that region.

To apply the above line of thought, it becomes imperative to first detect such regions in PO solutions that are separated by transition points. This is discussed in the next section.

8.1.1 Identifying Regions Separated by Transition Points (Active Set Partitioning)

Transition points may appear in the PO fronts of MOO problems usually because of two reasons, which are;

(a) Making a constraint active or inactive, or

(b) A sudden change in the nature of objective function(s).

The former is a more common cause of transition points and is addressed in this work. Recall from Section 8.1 that the transition point encountered in the truss problem was because of meeting the upper bound for variable x_2. The latter is not only less common in practice but also needs a more subtle mathematical treatment to first define what is meant by a "sudden" change in objective function(s). Although change in nature of any function can be defined in terms of higher order derivatives of a function, but it is currently not been studied in this work.

For every solution in a population of non-dominated solutions, we can find out which constraints are active and which ones are inactive. Then we can partition the solutions into groups based on proximity to different constraint boundaries. If all the transition points in a problem are a result of meeting some constraint bounds and not any other reason, then such a partitioning is all we want, to be able to learn the right rules for solutions of each partition rather than trying to learn a single rule for all of the solutions.

Consider a MOO problem with n_g inequality constraints $\{g_1, g_2, \ldots, g_{n_g}\}$ and n_x continuous variables $\{x_1, x_2, \ldots, x_{n_x}\}$. Without loss of generality, let all the inequality constraints be of *more than or equal to* type with corresponding upper bounds being g_1^u. These inequality

constraints can be represented as

$$\begin{aligned} & g_i \geq g_i^u \text{ or} \\ & g_i^n = \frac{g_i}{g_i^u} - 1 \geq 0, \end{aligned} \tag{8.4}$$

where g_i^n is the normalized constraint value for constraint g_i. Furthermore, let x_i^l and x_i^u where $i \in \{1, 2, \ldots, n_x\}$, represent the upper and lower bounds on the variables. In a manner similar to Equation (8.4), we can obtain constraints $\mathfrak{g}_1^l$ and $\mathfrak{g}_1^u$ which are given by

$$\begin{aligned} & \mathfrak{g}_i^l = \frac{x_i}{x_i^l} - 1 \geq 0 \text{ and} \\ & \mathfrak{g}_i^u = 1 - \frac{x_i}{x_i^u} \geq 0, \end{aligned} \tag{8.5}$$

where $\mathfrak{g}_i^l$ and $\mathfrak{g}_i^u$ are the normalized box constraints for variable x_i. In total, there are $n_c = n_g + 2 * n_x$ number of normalized constraints for this problem. Let `ctol` be the tolerance value such that if any of the normalized constraints are > 0 and $<=$ `ctol` then we call them as *active* constraints, else we call them *inactive* constraints. Let $h_i,\ i \in \{1, 2, \ldots, n_g\}$ be a boolean variable showing if the normalized constraint g_i^n is active or not by carrying a `1` or a `0` respectively. Similarly, $\mathfrak{h}_j^l,\ j \in \{1, 2, \ldots, n_x\}$ and $\mathfrak{h}_j^u,\ j \in \{1, 2, \ldots, n_x\}$ be boolean variables showing if the normalized constraints $\mathfrak{g}_i^l$ and $\mathfrak{g}_i^u$ are active or not.

With this information, let us look at an example to see how we can partition a set of solutions based on the active/inactive constraints for the solutions. Consider a MOO problem with one inequality constraint g_1 and two variables x_1 and x_2 respectively. Based on the aforementioned notation, the normalized constraints for this problem are g_1^n, $\mathfrak{g}_1^l$, $\mathfrak{g}_1^u$, $\mathfrak{g}_2^l$ and $\mathfrak{g}_2^u$. The corresponding boolean variables that show if for a solution, some constraint is active or not, are h_1^n, $\mathfrak{h}_1^l$, $\mathfrak{h}_1^u$, $\mathfrak{h}_2^l$ and $\mathfrak{h}_2^u$.

Table 8.1: An example of partitioning of solution space based on constraint activity.

Solution ID	h_1^n	$\mathfrak{h}_1^l$	$\mathfrak{h}_1^u$	$\mathfrak{h}_2^l$	$\mathfrak{h}_2^u$	Partition ID
1	1	0	1	0	0	5
2	1	0	1	0	0	5
3	1	0	1	0	0	5
4	1	0	0	1	0	9
5	1	0	0	1	0	9
6	0	0	0	1	0	8
7	0	0	0	1	0	8
8	0	0	0	1	0	8
9	0	0	0	0	0	0
10	0	0	0	0	0	0

Table 8.1 shows an example of ten non-dominated solutions for such a hypothetical MOO problem. It also shows the constraint activity for these ten solutions with the boolean variables h_1^n, $\mathfrak{h}_1^l$, $\mathfrak{h}_1^u$, $\mathfrak{h}_2^l$ and $\mathfrak{h}_2^u$. For a solution, if a boolean variable is 1, it means that the corresponding constraint is active for the solution. For example, a value of $h_1^n = 1$ means that the constraint g_1^n is active, otherwise its inactive. The boolean string for every solution can then be converted to its corresponding decimal value. For example, the boolean string for Solution ID-1 in the table is {1,0,1,0,0}. The corresponding decimal value for this string is calculated as:

$$1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 0 \cdot 2^4 = 5$$

We choose to call this decimal value as *partition ID* because it is unique for a set of active/inactive constraints. The solutions shown in Table 8.1 can be partitioned into four groups. Three solutions belong to partition ID-5, two belong to partition ID-9, three belong to partition ID-8 and two belong to partition ID-0. The rule learning method discussed in Section 7.1.3 needs some minimum number of solutions to learn meaningful rules. Hence, we can additionally define a minimum number of solutions to be present in a partition for it

to be considered for rule learning. All this is encapsulated in Algorithm 8.1. Let us look at

Algorithm 8.1 activeSetPartition( )

input: $\mathcal{S}$, `ctol`, `minPartSize` Non-dominated set, Constr. tol., minimum partition size
output: $\mathcal{P}$, $\mathcal{P}^v$ Partition IDs, Partitioned validity

1: $\mathcal{P}, \mathcal{P}^v \leftarrow \emptyset$
2: **for** $k \leftarrow 1$ **to** $|\mathcal{S}|$ **do**
3: $\quad s \leftarrow \mathcal{S}_k$
4: $\quad \mathbf{h} \leftarrow$ getBool(s,`ctol`) $\mathbf{h} = \{h_1, \ldots, h_{ng}, h_1^l, \ldots, h_{nx}^l, h_1^u, \ldots, h_{nx}^u\}$
5: $\quad p \leftarrow$ getPartitionId($\mathbf{h}$)
6: $\quad \mathcal{P} \leftarrow \mathcal{P} \frown p$ append to set $\mathcal{P}$
7: $[\mathcal{P}^u, \mathcal{P}^c] \leftarrow$ getUniqueCount($\mathcal{P}$)
8: **for** $i \leftarrow 1$ **to** $|\mathcal{P}|$ **do**
9: $\quad p \leftarrow \mathcal{P}_i$
10: $\quad$ **for** $j \leftarrow 1$ **to** $|\mathcal{P}^u|$ **do**
11: $\quad\quad u \leftarrow \mathcal{P}_j^u$
12: $\quad\quad c \leftarrow \mathcal{P}_j^c$
13: $\quad\quad$ **if** $(u == p) \wedge (c \geq$ `minPartSize`$)$ **then**
14: $\quad\quad\quad v \leftarrow$ `1`
15: $\quad\quad$ **else**
16: $\quad\quad\quad v \leftarrow$ `0`
17: $\quad\quad \mathcal{P}^v \leftarrow \mathcal{P}^v \frown v$

a line wise explanation of this algorithm.

- Input: The algorithm receives the non-dominated solutions set $\mathcal{S}$ for a MOO problem, minimum normalized constraint tolerance `ctol` for some constraint to be considered active/inactive at a solution and `minPartSize`, which is the minimum number of members in a partition for it to be considered a valid partition.

- Output: Output contains the partition ID $\mathcal{P}$ and partition validity $\mathcal{P}^v$, both of which have the same size as $\mathcal{S}$. The set $\mathcal{P}$ carries information on the partition ID of each solution of $\mathcal{S}$ and set $\mathcal{P}^v$ carries information on validity of the partition corresponding to each solution.

- Lines 1-6: Sets $\mathcal{P}$ and $\mathcal{P}^v$ are initialized to null sets. The For each non-dominated

solution $s \in \mathcal{S}$, we obtain the boolean string carrying information on which constraints are active and which ones are inactive for solution s based on the constraint values and the `ctol` parameter. This boolean string $\mathbf{h}$ is then converted into a decimal value p, which we are calling the partition ID to which solution s belongs. Then, the value of p is appended to the set $\mathcal{P}$.

- Line 7: Of the set of partition IDs obtained, the unique partition IDs along with their respective count are stored in $\mathcal{P}^u$ and $\mathcal{P}^c$ respectively.

- Lines 8-17: For each solution $s \in \mathcal{S}$, the validity of its partition v, is ascertained based on whether the corresponding partition ID p has at least `minPartSize` number of copies. The partition validity values of each solution are appended and stored in set $\mathcal{P}^v$.

Once such *active set partitioning* is achieved, both rule learning as well as variable repair based on those rules, is done individually for each partition. We can apply the rule learning and variable repair methods described in Section 7.1 to MOO problems having transition points if we learn rules (Section 7.1.3) and then repair variables (Section 7.1.5) of individual solutions of each partition separately. Let us now look at some results based on this idea.

8.1.2 Results on Problems with Transition Points

In this section, we present results for slightly modified versions of the two engineering design problems of Section 7.5. In each problem, the modification amounts to removal of some constraint that was earlier allowing us to get PO solutions without any transition points. In each problem;

- EMO/I learns the rules for each *active set partition* (ASP) separately and then attempts to repair the solutions using corresponding rules and,
- The repair strategy "Short Rules, No Preference on Variables" or SN from Table 7.2 is used.

In the text, we refer to this strategy of combining SN repair strategy with active set partitioning as 'SNasp'. We compare SNasp against repair strategies SN and NI (refer Table 7.2) in the results. The EMO/I parameters used for the three problems are shown in Table 8.2.

Table 8.2: EMO/I parameters used for solving the modified engineering design problems that contain transition points in the PO front.

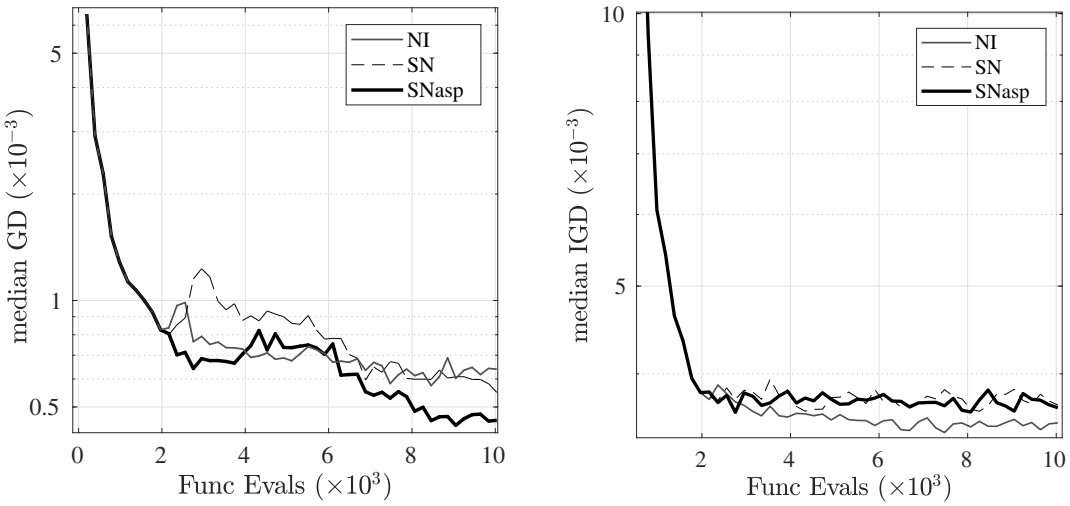
	TBT-2	MC-2
Population Size	52	200
Max Func. Evals	10,036	100,000
Prob. of Crossover	0.9	0.9
Prob. of Mutation	1/3	1/3
Crossover Index	10	10
Mutation Index	50	50
Start of Learning (% of Max FEs)	20	20
`ctol`	0.001	0.01
`minPartSize`	5% of pop	5% of pop

8.1.2.1 Modified Two Bar Truss Problem (TBT-2)

This problem is obtained from problem description given in Equation (7.23) by removing the constraint $g_2 \equiv V_2 \leq 0.044$ from the problem. This leads to the induction of a transition point in the PO front as shown in Figures 8.1 and 8.2.

Figure 8.3 shows that the SNasp repair strategy does better than both SN and NI strategy in terms of GD, whereas, because of presence of transition point, the SN repair strategy is unable to do better than NI strategy. The WRS results in Table 8.3 show that this difference

in performance of SNasp and SN is indeed significant. In case of IGD, both SN and SNasp perform inferior to NI. This is understandable as the repair strategies are aimed at improving convergence to the PO front and not convergence and diversity at the same time.



(a) GD results. (b) IGD results

Figure 8.3: Median GD and IGD results for TBT-2 problem over 30 runs.

Table 8.3: Results of Wilcoxon rank sum test for GD of TBT-2 problem comparing EMO/I repair strategies NI, SN and SNasp at 10,036 function evaluations and 5% significance level.

		Hypothesis	
		$\mathcal{H}_{\text{SNasp}\prec\text{NI}}$	$\mathcal{H}_{\text{SNasp}\prec\text{SN}}$
GD	h	Yes	Yes
	p	2.26×10^{-4}	0.047

8.1.2.2 Modified Metal Cutting Problem (MC-2)

This problem is obtained from problem description given in Equation (7.24) by removing the constraint $g_4 \equiv T_p(\mathbf{x}) \leq 0.9545$ from the problem. This leads to the induction of a transition point in the PO front as shown in Figures 8.4 and 8.5. The transition point divides the PO solutions into two regions shown in red and blue. The solutions of the two regions adhere to different set of rules as shown in Figure 8.5.

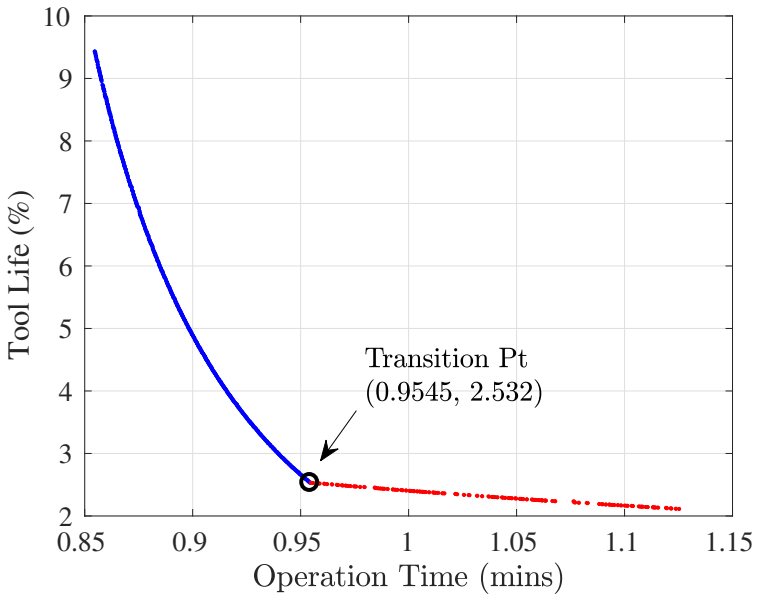


Figure 8.4: Transition point encountered in Metal Cutting problem shown in objective space.

Figure 8.6 shows that the SNasp repair strategy does better than both SN and NI strategy in terms of GD whereas because of presence of transition point, the SN repair strategy is not able to do better than NI strategy. The WRS results in Table 8.4 show that this difference in performance of SNasp and SN is indeed significant. In terms of GD, although Figure 8.6a shows that strategy SN performs better than NI, but WRS results in Table 8.4 do not back this observation which means the difference is not statistically significant.

In case of IGD, although Figure 8.6b shows that relative difference between the SNasp, SN and NI repair strategies, but the WRS results in Table 8.4 do not back this observation meaning that in term of IGD, the apparent difference between the three strategies is statistically insignificant.

Now that we have studied one way of handling MOO problems with transition points, let us now look at another idea of learning power laws involving functions of design variables as basis functions and then repairing variables based on those rules.

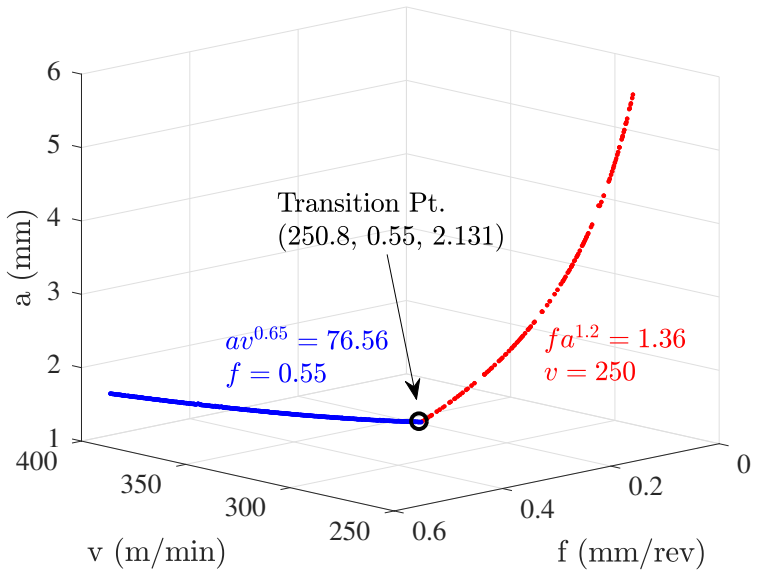


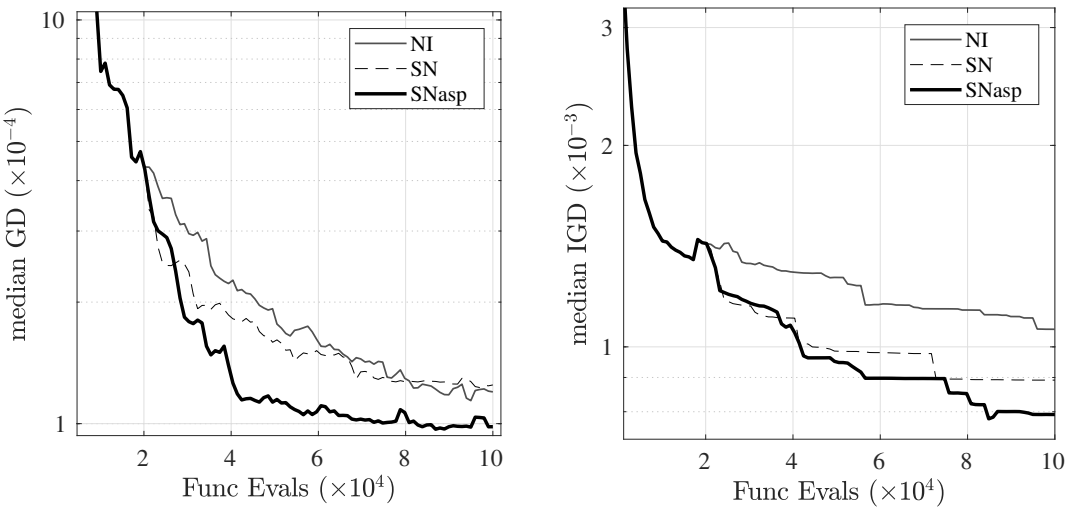
Figure 8.5: Transition point encountered in Metal Cutting problem shown in variable space.

Table 8.4: Results of Wilcoxon rank sum test for GD and IGD of MC-2 problem at 100,000 function evaluations and 5% significance level.

		Hypothesis		
		$\mathcal{H}_{\text{SNasp} \prec \text{NI}}$	$\mathcal{H}_{\text{SNasp} \prec \text{SN}}$	$\mathcal{H}_{\text{SN} \prec \text{NI}}$
GD	h	Yes	Yes	No
	p	0.0082	0.0306	-
IGD	h	No	No	No
	p	-	-	-

8.2 Power Law Rule Involving Functions of Variables

In Section 7.1, we looked at ways of learning fixed form rules involving variables only from non-dominated solutions and then repairing the non-dominated solutions based on those learned rules. Whether we made a repair based on a constant rule (see Section 7.1.5.1) or based on power law rule (see Section 7.1.5.2), making the repair was straight forward because the functions being learned were explicit in terms of all its constituent variables. If the learned rule is an implicit function of the variable that needs to be changed for solution repair, then this task may not be very straight forward. For example, if we learn a fixed form



(a) GD results.

(b) IGD results

Figure 8.6: Median GD and IGD results for MC-2 problem over 30 runs.

rule based on problem objectives or constraint functions or if we use CGP from Chapter 4 or Chapter 6 to find an algebraic expression involving problem variables that fits the data of non-dominated solutions but is implicit in terms of variable to be repaired.

Consider a bi-objective optimization problem with n_x variables given in Equation (8.6).

$$\begin{aligned} \text{Minimize} \quad & f_1(\mathbf{x}) \\ & f_2(\mathbf{x}) \\ \text{Subject to} \quad & g_i(\mathbf{x}) \leq 0 \; \forall i \in \{1, 2, \ldots, n_g\} \\ \text{Where} \quad & \mathbf{x} = \{x_1, x_2, \ldots, x_{n_x}\}, \\ & x_j \in [x_j^l, x_j^u] \; \forall \; j \in \{1, 2, \ldots, n_x\}, \text{ and} \\ & x_j \in \mathbb{R} \; \forall \; j. \end{aligned} \tag{8.6}$$

If we use f_1 and f_2 as our basis functions for learning power laws, we may learn functions of the form

$$f_1 \cdot f_2^b = c \tag{8.7}$$

from data of the non-dominated solutions. Estimating the parameters b and c is same as explained in Section 7.1.3.2. However, for repairing variables, Equation (8.7) can not be used in the same way a power law among variables could be used as described in Section 7.1.5.2. This is because the left hand side of Equation (8.7) may no longer be assumed to be an explicit expression in terms of the problem variables. In this section, we discuss a way we can use a rule that is implicitly defined in terms of the variables to make variable repairs.

Recall from Section 3.1 that in an MOO problem, the basis functions can be the design variables or functions thereof including the objectives and the constraints. Let us re-write Equation (8.7) as

$$\mathfrak{f}(\mathbf{x}) = \mathfrak{f}(x_1, x_2, \ldots, x_{n_x}) = c \tag{8.8}$$

where $\mathfrak{f}(\mathbf{x}) = f_1(\mathbf{x}) \cdot (f_2(\mathbf{x}))^b$. Since data of non-dominated solutions is used to arrive at this equation, in effect it is an implicit equation (in terms of problem variables) that describes the dominance relation.

Without loss of generality, let us say that we wish to repair variable x_r in an individual $\mathcal{I}$ of the non-dominated set. Let the individual $\mathcal{I}$ be represented as

$$\mathbf{x}_{\mathcal{I}} = \{a_1, a_2, \ldots, a_r, \ldots, a_{n_x}\} \tag{8.9}$$

in the variable space. The candidate individual, say $\mathcal{I}^+$, for repair can then be represented as

$$\mathbf{x}_{\mathcal{I}^+} = \{a_1, a_2, \ldots, x_r, \ldots, a_{n_x}\} \tag{8.10}$$

because only variable x_r of individual $\mathcal{I}$ is under consideration for repair. The problem of

repairing variable x_r can be turned into a one variable minimization problem as

$$\begin{aligned} \text{Minimize} \quad & | \mathfrak{f}(\mathbf{x}_{\mathcal{I}^+}) - c \,| \\ \text{Subject to} \quad & x_r \in [x_r^{l+}, x_r^{u+}]. \end{aligned} \tag{8.11}$$

Note that the bounds of x_r in Equation (8.11) are not the original bounds of variable x_r in Equation (8.6). The bounds $[x_r^{l+}, x_r^{u+}]$ are different from $[x_r^l, x_r^u]$ and represent the updated bounds of variable x_r derived from the non-dominated set utilized in learning the rule of Equation (8.7). This helps in efficiently reducing the search space. The single variable unconstrained minimization problem of Equation (8.11) can be solved using an inexpensive solver such as *Golden Section* method [111]. We can use $x_r = a_r$ as a starting point for the algorithm. Such a method is be a computationally inexpensive way of improving the performance of a non-dominated solution. It is true that the repaired individual $\mathcal{I}^+$ may turn out to infeasible because of not considering problem constraints in Equation (8.11), but we are expecting this to happen less often as we are already learning the rule from a non-dominated solutions set. In the next section, we see how this method performs on a simple engineering design problem.

8.2.1 Results on Truss problem

To test the method described in the previous section, we choose the two bar truss problem given by Equation (7.23). This is the problem without any transition points in the PO front. Furthermore, from [47] we know that the following relation exists between the two objectives in the PO solutions,

$$V \cdot S = 400. \tag{8.12}$$

To avoid wrongly attributing improvements in convergence to repairs based on rules involving variables only, which we know from Section 7.6.1 that they work, we decided to use only objectives as basis functions for these results. Again, the rule repair strategy SN (refer Table 7.2) was used. Only this time, the learned rules involved objective functions only. Once an individual is chosen for repair, we used Golden Section method to repair one of the chosen variables by solving the minimization problem given by Equation (8.11). Every golden section search was limited to a maximum of 100 iterations or convergence tolerance of 10^{-6}, whichever was reached earlier. We call this repair strategy as 'SNobj', i.e. learning and repairing based on rules involving objective functions only along with SN repair strategy.

Table 8.5 shows the other EMO/I parameters used in the work.

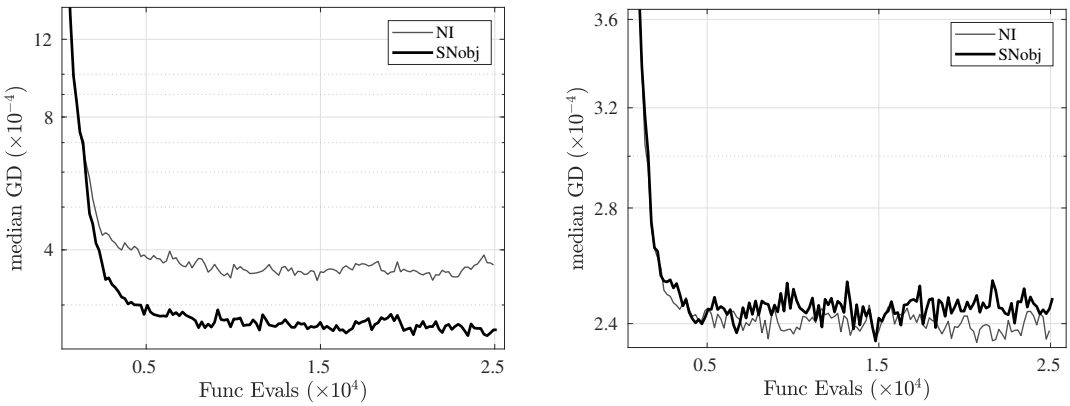
Table 8.5: EMO/I parameters used for solving the truss problem when only rules involving objective functions are learned.

Parameters	TBT-2
Population Size	92
Max Func. Evals	25,024
Prob. of Crossover	0.9
Prob. of Mutation	1/3
Crossover Index	10
Mutation Index	50
Start of Learning (% of Max FEs)	5

Table 8.6: Results of Wilcoxon rank sum test for GD of truss problem comparing EMO/I repair strategies NI and SNobj strategies at 25,024 function evaluations and 5% significance level.

		Hypothesis $\mathcal{H}_{\text{SNobj} \prec \text{NI}}$
GD	h	Yes
	p	4.5617×10^{-13}

Figure 8.7a shows that this strategy of learning only rules involving objective functions and then making repairs based on golden section method indeed helps the algorithm converge



(a) GD results. (b) IGD results

Figure 8.7: Median GD and IGD results for the Truss problem over 30 runs when only rules involving objective functions are learned.

faster to the PO front. This is backed by the WRS test results shown in Table 8.6. In terms of IGD, there is not much difference between the repair strategies SNobj and NI.

8.3 Concluding Remarks

In this chapter, we then looked at a way of handling transition points in the PO front by using active set partitioning method. Towards the end, we also discussed as to how we can also learn and make solution repairs based on rules that are implicitly defined in terms of the repairing variable using an inexpensive Golden Section method.

Chapter 9

Conclusion and Future Studies

This thesis identified and developed methods that can help with extracting human interpretable knowledge from EMO algorithms, while solving a MOO problem and using the same knowledge to repair the EMO solutions to expedite the algorithm.

The first half of this dissertation begins by showing a couple of fixed form rules that are prevalent in MOO engineering design problems and ways in which we can learn then on the fly during an EMO run in a more efficient manner than what is used by [11]. We then developed a customized GP that can be used for symbolically regressing rules from data as well as identifying decision boundary in binary classification problems, while the learned rules are constrained to be in the form of mathematical algebraic expressions involving regressands (features in case of classification) to maintain easy interpretability of the rules where we use the following definition of interpretability of rules/models, "how consistently can a human predict the results of the models" [112].

The use of a bi-objective optimization approach – minimization of classification error and minimization of rule complexity – has enabled us to find not one, but multiple rule structures having a trade-off between the two objectives. This allows an user to analyze a number of alternate trade-off rule structures for choosing a single or multiple of them for practice. In case of the industry problem, we used basic mathematical functions such as integration, differentiation and Fourier transform to arrive at a set of features from time

series manufacturing data. This is very different from the standard painstaking method of handcrafting features where each feature is created using an experts knowledge [113].

Our dimensional awareness procedure has been able to convert some of the obtained trade-off rules to dimensionally meaningful rules by using relevant problem constants. This technology allows a user to have a much better understanding and insights to the underlying process producing the data. The dimensional analysis along with searching for a knee solution provides the user with an effective decision support system when choosing from a PO set of regressors or classifiers obtained as a result of CGP. Both these ideas can be automated as well when we are following rule learning in an EMO with a rule based repair.

The second half of this dissertation developed methods of expediting convergence in EMO algorithms by using the aforementioned learned rules for making direct variable repairs during an EMO algorithm run to expedite its convergence. Unlike [71, 72], which learn rules in the form of decision trees and disjunctive normal form logical rules and add them as constraints to guide the algorithm, we learn rules in the form of algebraic expressions that can be used for direct variable repair to expedite the algorithm. A total of nine variable repair strategies for rule and variable selection were tested on test problems and engineering design problems.

The results showed that when faced with multiple qualifying rules to choose from, we should choose the shorter rules for making repairs. We did not see any effect of choosing variable (for repair) based on its frequency in the qualifying rules set. This was followed by developing method to learn and repair solutions in the presence of transition points in the PO front. Furthermore, we also proposed a computationally inexpensive method using of repairing variables when the learned rule is implicitly defined in terms of the variable to be repaired.

9.1 Contributions of this Thesis

The key contributions of this thesis, presented in the order of appearance in this thesis, can be summarized as follows:

1. We developed a custom GP for a symbolic regression task that can learn rules from data in form of algebraic expressions. This custom GP has many interesting ideas implemented from the literature such as, bi-objective formulation to control bloat, multiple small expression trees instead of a single expression tree to make genetic operations on expression trees more efficient, weights learning using a faster classical method such as OLSR to learn constants in the rule much more efficiently as compared to regular GPs. We also developed a custom diversity preserving mechanism in GP that penalizes duplicate solutions in the population while maintaining the relative non-domination rank order between the penalized duplicates belonging to different non-domination ranks.

2. We developed a custom dimensional inconsistency penalty metric that can calculate the level of dimensional inconsistency in an algebraic expression. This metric is very helpful in identifying PO set of rules learned by CGP that are not dimensionally meaningful.

3. We developed a custom GP for classification task that can learn decision boundary between two classes as an algebraic expression using the problem features. This custom GP performed well on binary class data from industry. The CGP could provide multiple PO rules with varying trade-off between classification error and rule complexity. It could also select the six most important features out of more than fifty features to build the classifiers. These important features were in line with the understanding of the process of our industry partners which they obtained by conducting physical

experiments and analytic studies of the same process in a controlled environment. Furthermore, the knowledge of dimensional consistency of these classifiers was found to be beneficial for planning future experiments studies to understand the physics of the process.

4. We developed the EMO/I framework that combines the idea of innovization with that of an EMO algorithm. We could evaluate various repair strategies when repairing solutions based on multiple rules in the form of algebraic expressions. Results on test problems and engineering design problems show that choosing smaller rules first from qualifying rules for repair is more beneficial for faster convergence of EMO.

5. We developed a custom methodology, *active set partitioning*, to address the issue of learning rules and repairing solutions in the presence of transition points in the PO front. Furthermore, a computationally efficient method of repairing solutions when a rule is defined implicitly in terms of the variables is also suggested, which has been shown to work on an engineering design problem.

9.2 Future Studies

This study has opened doors for many exciting ideas/questions to be explored next. A few of them are mentioned below:

9.2.1 GP with in Tandem Dimensional Consistency Check

In Chapters 4 to 6, we have shown how a dimensional consistency check on the final PO solutions of CGP is a powerful tool to shortlist dimensionally meaningful rules. Instead of

using this check at the end of CGP, i.e. serially, it may be beneficial to use it during the run of CGP. A few ways to use this dimensional inconsistency penalty are:

- Modifying the tree crossover operator in CGP to avoid crossover operation between trees whose dimensional inconsistency penalties are very different. Similarly, making sure not to mutate trees that have zero dimensional inconsistency penalty.
- Using the dimensional inconsistency penalty value in survivor selection operation of the CGP instead of the currently used crowded tournament selection operator [31].

9.2.2 Non-linear Decision Trees

Figure 9.1 shows the idea of combining CGP with that of decision tree based classifier to capture complex and even disconnected decision boundaries in the feature space. The figure shows an arbitrary binary classification problem where the green patches represent *Go* class data and the red color data represents the *NoGo* class data. If one uses only a decision tree

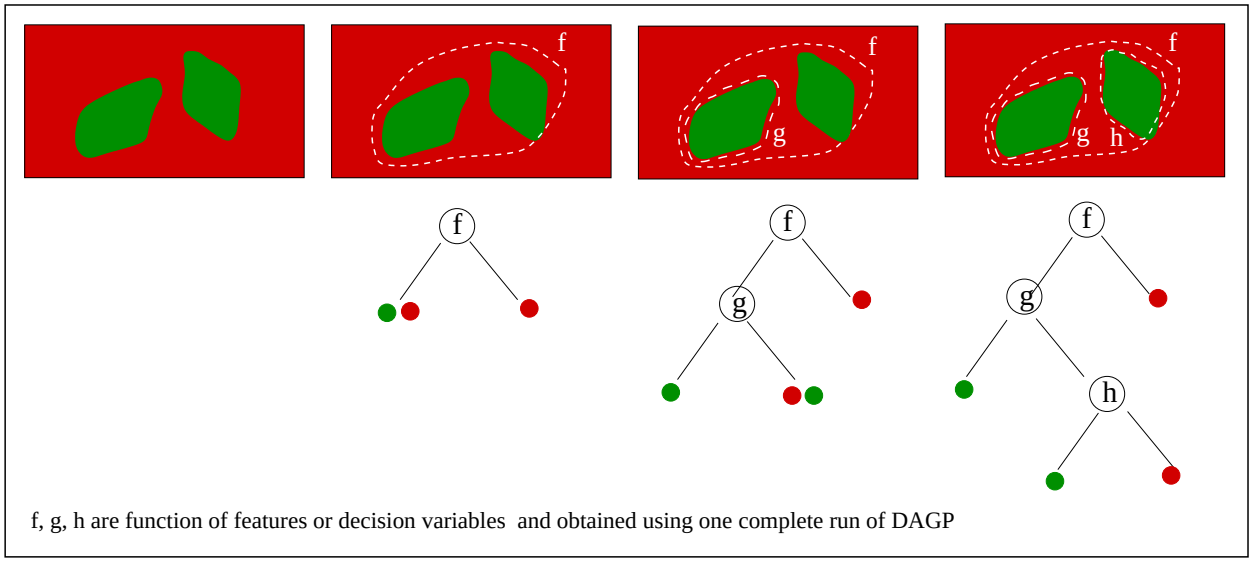


Figure 9.1: The idea of combining CGP with a Decision Tree classification algorithm.

based classifier in this problem, the decision tree obtained will be very deep as a decision tree

tries to partition the feature space parallel to the coordinate axes representing the features. However, when combined with CGP such that each decision node of the decision tree is obtained using a CGP run minimizing node impurity, the final tree will be a smaller tree with each node of the tree being an algebraic expression of original features. Such a method can be utilized to derive a classifier that is both effective and interpretable.

9.2.3 Experimenting with Different Definitions of Rule Complexity

The complexity aspect of a learned model can be defined in many ways [94]. In CGP, we defined complexity of a CGP individual simply as the number of nodes in the corresponding expression tree. It can also be calculated differently if we assign various operators in the function set of CGP different complexities based on some a-priori hierarchy. For example, the set of operations $\{+, -, \div, \times\}$ can be assigned a lower complexity value than say $\{\sqrt{}, exp(), \log()\}$ functions. These hierarchies can be based on user's preference and knowledge to aid the CGP in finding the appropriate rules from data.

9.2.4 Measuring Efficacy of Repairs

In the repair methodology chosen in Chapters 7 and 8, once at least one qualifying rule is found to make repairs, an extra population with same size as the population size of EMO/I is created that carries the repaired population. This may be inefficient use of the computational budget as we get limited number of evaluations of the objective functions. However, if we start measuring the efficacy of repair operation in every generation and then, keep reducing/increasing the size of population for repair from previous generation based

on the degradation/improvement of this repair efficacy metric, we may be able to use our computational budget more efficiently. One metric to measure the efficacy of the repair operation is the fraction of population members present in each generation that came via a repair operation instead of any of the genetic operations.

Finally, we should apply the methods developed in this thesis to more real world problems to make it more generally applicable. The ideas developed in this thesis can be very useful for practitioners in MOO as well as machine learning. I hope that this thesis contributes, in howsoever small way, towards bridging the gap between human expertise in optimal design and its digital analog.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Siemens-PLM-HEEDS, "We help you discover better designs, faster," https://www.plm.automation.siemens.com/global/en/products/simcenter/simcenter-heeds.html, 2004, [Online].

[2] Esteco-modeFrontier, "Process automation and optimization in the engineering design process," https://www.esteco.com/modefrontier, 1998, [Online].

[3] Dassault-iSight, "Automate Design Exploration and Optimization," https://www.3ds.com/products-services/simulia/products/isight-simulia-execution-engine/, 1996, [Online].

[4] Noesis-Optimus, "The Industry-Leading PIDO Software Platform," https://www.noesissolutions.com/our-products/optimus/, 2008, [Online].

[5] Dynardo-optiSLang, "Robust Design Optimization (RDO) in virtual product development," https://www.dynardo.de/en/software/optislang.html, 2002, [Online].

[6] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen *et al.*, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007, vol. 5.

[7] H. J. Levesque, "Knowledge representation and reasoning," *Annual review of computer science*, vol. 1, no. 1, pp. 255–287, 1986.

[8] R. Davis, H. Shrobe, and P. Szolovits, "What is a knowledge representation?" *AI magazine*, vol. 14, no. 1, pp. 17–17, 1993.

[9] D. P. Bertsekas, A. Nedi, A. E. Ozdaglar *et al.*, *Convex analysis and optimization*. Athena Scientific, 2003.

[10] K. Deb and A. Srinivasan, "Innovization: Innovating design principles through optimization," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006, pp. 1629–1636.

[11] S. Bandaru and K. Deb, "Automated discovery of vital knowledge from pareto-optimal solutions: First results from engineering design," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–8.

[12] ——, "Higher and lower-level knowledge discovery from pareto-optimal sets," *Journal of Global Optimization*, vol. 57, no. 2, pp. 281–298, 2013.

[13] A. Ng, K. Deb, and C. Dudas, "Simulation-based innovization for production systems improvement: An industrial case study," in *Proceedings of The International 3rd*

Swedish Production Symposium, SPS'09, Goteborg, Sweden, 2-3 December 2009. The Swedish Production Academy, 2009, pp. 278–286.

[14] S. Bandaru, A. H. Ng, and K. Deb, "Data mining methods for knowledge discovery in multi-objective optimization: Part a-survey," *Expert Systems with Applications*, vol. 70, pp. 139–159, 2017.

[15] I. Nonaka and H. Takeuchi, *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford university press, 1995.

[16] A. Inselberg, "The plane with parallel coordinates," *The visual computer*, vol. 1, no. 2, pp. 69–91, 1985.

[17] A. M. Geoffrion, J. S. Dyer, and A. Feinberg, "An interactive approach for multi-criterion optimization, with an application to the operation of an academic department," *Management science*, vol. 19, no. 4-part-1, pp. 357–368, 1972.

[18] K. J. Chichakly and M. J. Eppstein, "Discovering design principles from dominated solutions." *IEEE Access*, vol. 1, no. iii, pp. 275–289, 2013.

[19] S. Obayashi and D. Sasaki, "Visualization and data mining of pareto solutions using self-organizing map," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2003, pp. 796–809.

[20] A. H. Ng, C. Dudas, J. Nießen, and K. Deb, "Simulation-based innovization using data mining for production systems analysis," in *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*. Springer, 2011, pp. 401–429.

[21] T. W. Simpson, J. Poplinski, P. N. Koch, and J. K. Allen, "Metamodels for computer-based engineering design: survey and recommendations," *Engineering with computers*, vol. 17, no. 2, pp. 129–150, 2001.

[22] B. Kim, R. Khanna, and O. O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," in *Advances in Neural Information Processing Systems*, 2016, pp. 2280–2288.

[23] M. E. Newman, "Power laws, pareto distributions and zipf's law," *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.

[24] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[25] C. Molnar, *Interpretable Machine Learning*, 2019, accessed: 2019-04-16.

[26] P. W. Bridgman, *Dimensional analysis*. Yale university press, 1922.

[27] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic programming: an introduction*. Morgan Kaufmann San Francisco, 1998, vol. 1.

[28] C. Myburgh and K. Deb, "Derived heuristics-based consistent optimization of material flow in a gold processing plant," *Engineering optimization*, vol. 50, no. 1, pp. 1–18, 2018.

[29] K. Deb and R. Datta, "Hybrid evolutionary multi-objective optimization and analysis of machining operations," *Engineering Optimization*, vol. 44, no. 6, pp. 685–706, 2012.

[30] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2012, vol. 821.

[31] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[32] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[33] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[34] H.-P. P. Schwefel, *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[35] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[36] C. A. C. Coello, D. A. VanVeldhuizen, and G. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Boston, MA: Kluwer, 2002.

[37] S. M. Sait, H. Youssef, and J. A. Khan, "Fuzzy evolutionary algorithm for vlsi placement," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2001, pp. 1056–1063.

[38] J. E. Rodríguez, A. L. Medaglia, and J. P. Casas, "Approximation to the optimum design of a motorcycle frame using finite element analysis and evolutionary algorithms," in *2005 IEEE Design Symposium, Systems and Information Engineering*. IEEE, 2005, pp. 277–285.

[39] Y.-J. Kim and J. Ghaboussi, "A new genetic algorithm based control method using state space reconstruction," in *Proceedings of the Second World Conference on Struc. Control*, 1998, pp. 2007–2014.

[40] K. Harada, K. Ikeda, and S. Kobayashi, "Hybridization of genetic algorithm and local search in multiobjective function optimization: recommendation of ga then ls," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006, pp. 667–674.

[41] S. P. Harris and E. C. Ifeachor, "Nonlinear fir filter design by genetic algorithm," in *Proceedings of the First Online Workshop on Soft Computing (WSC1)*, 1996, pp. 216–221.

[42] F. Y. Cheng and D. Li, "Multiobjective optimization design with pareto genetic algorithm," *Journal of Structural Engineering*, vol. 123, no. 9, pp. 1252–1261, 1997.

[43] M. S. Bright, "Evolutionary strategies for the high-level synthesis of vlsi-based dsp systems for low power," Ph.D. dissertation, University of Wales. Cardiff, 1998.

[44] A. Belegundu, E. Constans, R. Salagame, and D. Murthy, "Multi-objective optimization of laminated ceramic composites using genetic algorithms," in *5th Symposium on Multidisciplinary Analysis and Optimization*, 1994, p. 4363.

[45] 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*, vol. 15, pp. 42–56, 2014.

[46] M. W. Mkaouer, M. Kessentini, S. Bechikh, K. Deb, and M. Ó Cinnéide, "Recommendation system for software refactoring using innovization and interactive dynamic optimization," in *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*. ACM, 2014, pp. 331–336.

[47] K. Deb and A. Srinivasan, "Innovization: Discovery of innovative design principles through multiobjective evolutionary optimization," in *Multiobjective Problem Solving from Nature*. Springer, 2008, pp. 243–262.

[48] S. Bandaru, C. C. Tutum, K. Deb, and J. H. Hattel, "Higher-level innovization: A case study from friction stir welding process optimization," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 2782–2789.

[49] S. Bandaru, "Automated innovization: Knowledge discovery through multi-objective optimization," Ph.D. dissertation, Indian Institute of Technology-Kanpur, Dept. of Mechanical Engineering, India, 2012.

[50] 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, no. 5, pp. 1054–1074, 2009.

[51] S. Bandaru and K. Deb, "Temporal innovization: Evolution of design principles using multi-objective optimization," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2015, pp. 79–93.

[52] A. H. Ng, S. Bandaru, and M. Frantzén, "Innovative design and analysis of production systems by multi-objective optimization and data mining," *Procedia CIRP*, vol. 50, pp. 665–671, 2016.

[53] B. Meyer and K. Sugiyama, "The concept of knowledge in km: a dimensional model," *Journal of knowledge management*, vol. 11, no. 1, pp. 17–35, 2007.

[54] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley, "Dna visual and analytic data mining," in *Proceedings. Visualization'97 (Cat. No. 97CB36155)*. IEEE, 1997, pp. 437–441.

[55] I. Hatzilygeroudis and J. Prentzas, "Integrating (rules, neural networks) and cases for knowledge representation and reasoning in expert systems," *Expert Systems with Applications*, vol. 27, no. 1, pp. 63–75, 2004.

[56] J. Shawe-Taylor and N. Cristianini, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[57] M. H. Hassoun, *Fundamentals of Artificial neural networks*. Cambridge: MIT Press, 1995.

[58] G. Hinton and T. J. Sejnowski, *Unsupervised Learning: Foundations of Neural Computation*, 1st ed. MIT Press, 1999.

[59] N. Hitomi, H. Bang, and D. Selva, "Extracting and applying knowledge with adaptive knowledge-driven optimization to architect an earth observing satellite system," in *AIAA Information Systems-AIAA Infotech@ Aerospace*, 2017, p. 0794.

[60] B. L. W. H. Y. Ma, B. Liu, and Y. Hsu, "Integrating classification and association rule mining," in *Proceedings of the fourth international conference on knowledge discovery and data mining*, 1998, pp. 24–25.

[61] S. Bandaru and K. Deb, "Towards automating the discovery of certain innovative design principles through a clustering-based optimization technique," *Engineering optimization*, vol. 43, no. 9, pp. 911–941, 2011.

[62] A. H. Ng, C. Dudas, J. Nießen, and K. Deb, "Simulation-based innovization using data mining for production systems analysis," in *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*. Springer, 2011, pp. 401–429.

[63] N. R. Draper and H. Smith, *Applied regression analysis*. John Wiley & Sons, 2014, vol. 326.

[64] Y. Bernstein, X. Li, V. Ciesielski, and A. Song, "Multiobjective parsimony enforcement for superior generalisation performance," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, vol. 1. IEEE, 2004, pp. 83–89.

[65] E. D. De Jong and J. B. Pollack, "Multi-objective methods for tree size control," *Genetic Programming and Evolvable Machines*, vol. 4, no. 3, pp. 211–233, 2003.

[66] H. Iba, "Bagging, boosting, and bloating in genetic programming," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*. Morgan Kaufmann Publishers Inc., 1999, pp. 1053–1060.

[67] D. LLC. (2018) Eureqa : The a.i.-powered modeling engine by datarobot. [Online]. Available: https://www.nutonian.com/products/eureqa/

[68] C. W. Ahn and R. S. Ramakrishna, "On the scalability of real-coded bayesian optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 307–322, 2008.

[69] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Boa: The bayesian optimization algorithm," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*. Morgan Kaufmann Publishers Inc., 1999, pp. 525–532.

[70] A. Zhou, Q. Zhang, and Y. Jin, "Approximating the set of pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm," *IEEE transactions on evolutionary computation*, vol. 13, no. 5, pp. 1167–1189, 2009.

[71] A. H. Ng, C. Dudas, H. Boström, and K. Deb, "Interleaving innovization with evolutionary multi-objective optimization in production system simulation for faster convergence," in *Learning and Intelligent Optimization*. Springer, 2013, pp. 1–18.

[72] R. S. Michalski, G. Cervone, and K. Kaufman, "Speeding up evolution through learning: Lem," in *Intelligent Information Systems*. Springer, 2000, pp. 243–256.

[73] M. Keijzer and V. Babovic, "Dimensionally aware genetic programming," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*. Morgan Kaufmann Publishers Inc., 1999, pp. 1069–1076.

[74] A. Ratle and M. Sebag, "Genetic programming and domain knowledge: Beyond the limitations of grammar-guided machine discovery," in *Parallel Problem Solving from Nature PPSN VI*. Springer, 2000, pp. 211–220.

[75] R. I. Mckay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. Oneill, "Grammar-based genetic programming: a survey," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3-4, pp. 365–396, 2010.

[76] D. J. Montana, "Strongly typed genetic programming," *Evolutionary computation*, vol. 3, no. 2, pp. 199–230, 1995.

[77] M. Keijzer and V. Babovic, "Declarative and preferential bias in gp-based scientific discovery," *Genetic Programming and Evolvable Machines*, vol. 3, no. 1, pp. 41–79, 2002.

[78] A. Ashour, L. Alvarez, and V. Toropov, "Empirical modelling of shear strength of rc deep beams by genetic programming," *Computers & structures*, vol. 81, no. 5, pp. 331–338, 2003.

[79] S. Bandaru and K. Deb, "A dimensionally-aware genetic programming architecture for automated innovization," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2013, pp. 513–527.

[80] N. Padhye and K. Deb, "Multi-objective optimisation and multi-criteria decision making in sls using evolutionary approaches," *Rapid Prototyping Journal*, vol. 17, no. 6, pp. 458–478, 2011.

[81] K. Deb and K. Sindhya, "Deciphering innovative principles for optimal electric brushless dc permanent magnet motor design," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 2283–2290.

[82] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012, vol. 329.

[83] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[84] T. S. Jaakkola and D. Haussler, "Probabilistic kernel regression models." in *AISTATS*, 1999.

[85] J. R. Koza, *Genetic Programming II, Automatic Discovery of Reusable Subprograms*. MIT Press, Cambridge, MA, 1992.

[86] M. F. Brameier and W. Banzhaf, *Linear genetic programming*. Springer Science & Business Media, 2007.

[87] S. Silva and J. Almeida, "Gplab-a genetic programming toolbox for matlab," in *Proceedings of the Nordic MATLAB conference*. Citeseer, 2003, pp. 273–278.

[88] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *science*, vol. 324, no. 5923, pp. 81–85, 2009.

[89] P. J. Angeline, "Subtree crossover: Building block engine or macromutation," *Genetic programming*, vol. 97, pp. 9–17, 1997.

[90] E. Analytics. (2018) Datamodeler : by evolved analytics. [Online]. Available: http://www.evolved-analytics.com/?q=about

[91] M. Kuhn and K. Johnson, *Applied predictive modeling*. Springer, 2013, vol. 26.

[92] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[93] S. Bleuler, J. Bader, and E. Zitzler, "Reducing bloat in gp with multiple objectives," in *Multiobjective Problem Solving from Nature*. Springer, 2008, pp. 177–200.

[94] B.-T. Zhang and H. Mühlenbein, "Balancing accuracy and parsimony in genetic programming," *Evolutionary Computation*, vol. 3, no. 1, pp. 17–38, 1995.

[95] D. Searson, M. Willis, and G. Montague, "Co-evolution of non-linear pls model components," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 21, no. 12, pp. 592–603, 2007.

[96] J. Doe. (2014) Height, depth and level of a tree. [Online]. Available: http://typeocaml.com/2014/11/26/height-depth-and-level-of-a-tree/

[97] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman, *Applied linear statistical models*. Irwin Chicago, 1996, vol. 4.

[98] K. Deb and S. Gupta, "Understanding knee points in bicriteria problems and their implications as preferred solution principles," *Engineering optimization*, vol. 43, no. 11, pp. 1175–1204, 2011.

[99] B. R. Munson, T. H. Okiishi, A. P. Rothmayer, and W. W. Huebsch, *Fundamentals of fluid mechanics*. John Wiley & Sons, 2014.

[100] J. E. Shigley, C. R. Mischke, and R. G. Budynas, *Mechanical Engineering Design, International*. Mc-Graw Hill Book Co., Singapore, 1989.

[101] D. Chakraborty, S. Soni, J. Wei, N. Kovvali, A. Papandreou-Suppappola, D. Cochran, and A. Chattopadhyay, "Physics based modeling for time-frequency damage classification," in *Modeling, Signal Processing, and Control for Smart Structures 2008*, vol. 6926. International Society for Optics and Photonics, 2008, p. 69260M.

[102] J. Branke, K. Deb, H. Dierolf, and M. Osswald, "Finding knees in multi-objective optimization," in *International conference on parallel problem solving from nature*. Springer, 2004, pp. 722–731.

[103] G. Strang, G. Strang, G. Strang, and G. Strang, *Introduction to linear algebra*. Wellesley-Cambridge Press Wellesley, MA, 1993, vol. 3.

[104] G. Casella and R. L. Berger, *Statistical inference*. Duxbury Pacific Grove, CA, 2002, vol. 2.

[105] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[106] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 1322–1328.

[107] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[108] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[109] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[110] R. Q. Sardiñas, M. R. Santana, and E. A. Brindis, "Genetic algorithm-based multi-objective optimization of cutting parameters in turning processes," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 2, pp. 127–133, 2006.

[111] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Golden section search in one dimension," *Numerical Recipes in C: The Art of Scientific Computing*, p. 2, 1992.

[112] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[113] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: data mining, inference, and prediction, springer series in statistics," 2009.