Multiobjective Evolutionary Approaches to Pattern Classification and Robust Optimization

Thesis submitted by
Kaustuv Nag

Doctor of Philosophy (Engineering)

Department of Instrumentation and Electronics Engineering
Faculty Council of Engineering and Technology
Jadavpur University
Kolkata, India
2018

To my family...

INDEX NO. 77/12/E

1. Title of the Thesis: Multiobjective Evolutionary Approaches to Pattern Classification and Robust Optimization

2. Name, Designation, and Institute of the Supervisors:

i. Prof. Rajani Kanta Mudi
Professor
Department of Instrumentation and Electronics Engineering
Jadavpur University
Kolkata-700098

ii. Dr. Tandra Pal
Associate Professor
Department of Computer Science and Engineering
National Institute of Technology Durgapur
Durgapur-713209

iii. Prof. Nikhil Ranjan Pal
Professor
Electronics and Communication Sciences unit
Indian Statistical Institute
Calcutta-700108

3. List of Publications

i. Journal Publications

1. **Kaustuv Nag** and Nikhil Ranjan Pal, "Robustness in multiobjective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, (in review).
2. **Kaustuv Nag** and Nikhil Ranjan Pal, "Feature extraction and selection for parsimonious classifiers with multiobjective genetic programming," *IEEE Transactions on Evolutionary Computation*, (revision submitted).

3. **Kaustuv Nag**, Tandra Pal, Rajani Kanta Mudi, and Nikhil Ranjan Pal, "Robust multiobjective optimization with robust consensus," *IEEE Transactions on Fuzzy Systems*, 2018.
   DOI: 10.1109/TFUZZ.2018.2848261.
4. **Kaustuv Nag** and Nikhil Ranjan Pal, "A multiobjective genetic programming based ensemble for simultaneous feature selection and classification," *IEEE Transactions on Cybernetics*, vol. 46, no. 2, pp. 499-510, 2015.
   DOI: 10.1109/TCYB.2015.2404806.
5. **Kaustuv Nag**, Tandra Pal, and Nikhil Ranjan Pal, "ASMiGA: An archive-based steady-state micro genetic algorithm," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 40-52, 2014.
   DOI: 10.1109/TCYB.2014.2317693.
6. Prasenjit Dey, **Kaustuv Nag**, Tandra Pal, and Nikhil Ranjan Pal. "Regularizing multilayer perceptron for robustness," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 8, pp. 1255-1266, 2018.
   DOI: 10.1109/TSMC.2017.2664143.
7. Saurajyoti Kar, **Kaustuv Nag**, Abhishek Dutta, Denis Constales, and Tandra Pal, "An improved cellular automata model of enzyme kinetics based on genetic algorithm," *Chemical Engineering Sciences*, vol. 110, pp. 105-118, 2014.
   DOI: 10.1016/j.ces.2013.08.013.

ii. Conference Publications

8. **Kaustuv Nag**, Tandra Pal, and Nikhil Ranjan Pal, "Robust consensus: A new measure for multicriteria robust group decision making problems using evolutionary approach," *International Conference on Artificial Intelligence and Soft Computing*, pp. 384-394, Springer, 2014.
   DOI: 10.1007/978-3-319-07173-2_33.
9. **Kaustuv Nag** and Tandra Pal, "A new archive based steady state genetic algorithm," *2012 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1-7, IEEE, 2012.
   DOI: 10.1109/CEC.2012.6256448.

iii. Book Chapters

10. **Kaustuv Nag** and Nikhil Ranjan Pal, "Genetic programming for classification and feature selection," in *Evolutionary and Swarm Intelligence Algorithms.* Springer, 2019, pp. 119-141.
DOI: 10.1007/978-3-319-91341-4_7.

4. List of Patents: None

5. List of Presentations in International/National Conferences: None

Certificate from the Supervisors

This is to certify that the thesis entitled "**Multiobjective Evolutionary Approaches to Pattern Classification and Robust Optimization**" submitted by Kaustuv Nag, who got his name registered on October 9, 2012 for the award of Ph.D. (Engg.) degree of Jadavpur University, is absolutely based upon his own work under the supervision of Prof. Rajani Kanta Mudi, Dr. Tandra Pal, and Prof. Nikhil Ranjan Pal and that neither his thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

.....
(**Prof. Rajani Kanta Mudi**)

.....
(**Dr. Tandra Pal**)

.....
(**Prof. Nikhil Ranjan Pal**)

Acknowledgements

I am immensely indebted to my supervisors, Prof. Rajani Kanta Mudi, Dr. Tandra Pal, and Prof. Nikhil Ranjan Pal for their kind and invaluable supervision. I appreciate Department of Science and Technology, Ministry of Science and Technology, Govt. of India, for providing me financial support in the form of INSPIRE Fellowship (Code No. IF120686). I am thankful to all the faculty members, staffs, and students of the Department of Instrumentation and Electronics Engineering, Jadavpur University; Department of Computer Science and Engineering, National Institute of Technology Durgapur; and Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata. I am grateful to my friends. Special thanks go to Dr. Prasenjit Dey, Mr. Dipankar Kundu, and Dr. Amar Kishor. I want to acknowledge my family members, especially my parents and grandfather, for everything.

Kaustuv Nag

Abstract

In this thesis, we use multiobjective evolutionary approaches to address four research problems related to decision making. First, we propose a method for simultaneous feature selection (FS) and classification. In the next work, we propose a method of feature extraction and selection for designing parsimonious classifiers. In the succeeding work, in the context of fuzzy group decision making for multiobjective optimization problems, we propose a new measure, called robust consensus, and incorporate it in the search process. Finally, we propose a measure of sensitivity that can assess the sensitivity, and hence, robustness of a multiobjective solution. In the same work, we also propose three approaches to use this measure in the search process.

In the first work, we present an integrated algorithm for simultaneous FS and design of diverse classifiers using a steady state multiobjective genetic programming, which minimizes three conflicting objectives: false positives, false negatives, and the number of leaf nodes in the tree. Our method divides a c-class problem into c binary classification problems. It evolves c sets of genetic programs to create c ensembles. During mutation operation, our method exploits the fitness as well as the unfitness of features, which dynamically change with generations with a view to using a set of highly relevant features with low redundancy. The classifiers of the i^{th} class determine the *net belongingness* of an unknown data point to the i^{th} class using a weighted voting scheme, which makes use of the false positive and false negative mistakes made on the training data. We test our method on eight microarray and eleven text data sets with a diverse number of classes (from 2 to 44), a large number of features (from 2000 to 49151), and a high feature-to-sample ratio (from 1.03 to 273.1). We compare our method with a bi-objective genetic programming scheme that does not use any FS and rule size reduction strategy. It helps to depict the effectiveness of the proposed FS and rule size reduction schemes. Furthermore, we compare our method with four classification methods in conjunction with six features selection algorithms and the full feature set. Our scheme performs the best for 380 out of 474 combinations of data sets, classification algorithm and FS method.

The primary objective of the next work is to design classifiers that are reasonably simple and understandable yet capable of yielding a good performance. Thus, our

problem here is to obtain an ensemble of linear classifiers with as few simple features as possible. For this we use genetic programming to extract features that can linearly separate two classes. We propose an integrated mechanism for simultaneously extracting and selecting useful linearly separable features. We decompose a c-class problem into c binary classification problems and evolve c sets of binary classifiers employing a steady-state multiobjective genetic programming with three minimizing objectives. Each binary classifier is composed of a binary tree and a linear support vector machine (SVM). The features extracted by the feature nodes and some of the function nodes of the tree are used to train the SVM. The decision made by the SVM is considered the decision of the corresponding classifier. During crossover and mutation, the SVM-weights are used to determine the usefulness of the corresponding nodes. We also employ unfitness functions for the feature nodes and a fitness function based on Golub's index to select useful features. We compare our method with 34 classification systems using 18 data sets. The performance of the proposed method is found to be better than 432 out of 570, i.e., 75.79% of comparing cases. Our results confirm that the proposed method is capable of achieving our objectives.

In the succeeding work, we consider a multiobjective robust optimization problem where a set of weighted decision makers provides their preferences a priori. The preferences are provided either in the objective space or in the decision variable space using fuzzy numbers. To solve this problem the following three things are required: 1) an indicator to measure consensus, 2) an indicator to measure the robustness of the solutions to their degrees of consensus, and 3) a reformulation of the multiobjective robust optimization problem. It is necessary for the reformulated problem to generate robust solutions that also enjoy high degrees of consensus. In this work, we have addressed these three issues. For this purpose, we have proposed two approaches to define *consensus*. Then, we have extended these approaches to define *robust consensus*, an indicator to measure the robustness of a given solution to its degree of consensus. Though these approaches can be used to define a countless number of measures, we have proposed 12 definitions of consensus, and hence, robust consensus. Furthermore, we have proposed two ways for the reformulation. Experimental results illustrate that the behaviors of the proposed definitions and of the reformulations are consistent with our expectations.

In the final work, in the context of multiobjective optimization problems, we have introduced a new measure of sensitivity, which is inversely related to robustness of solutions. It quantifies the robustness of a solution with respect to perturbations in the variable space. We have shown how the cost of computing this measure can be

reduced using an approximation with the first-order Taylor series expansion subject to the following three conditions. First, the objective functions are differentiable. Second, the input noise is any one of uniform noise, additive white Gaussian noise, and multiplicative noise. Third, the changes in the variables are very small. Next, we have proposed three approaches to reformulate MOPs. When the first approach is used, solving the reformulated MOP yields solutions of the original MOP with different degrees of sensitivity/robustness. When the other two approaches are used, the reformulated MOPs yield solutions with sensitivities less than a given threshold. We have experimentally validated our claims and have analyzed the behaviour of the reformulation strategies.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction and the Scope of the Thesis

1.1 Introduction

In this thesis, we use multiobjective evolutionary approaches to address four problems related to decision making. Pattern classification is a straightforward task of decision making, where selection of useful features is essential. To address this issue, in the first work of this thesis, we propose an ensemble based method for simultaneous feature selection and classification using a multiobjective evolutionary approach. Especially, if we can extract and use features that can linearly separate a given data set, it may increase the parsimony and the interpretability of the classifier. In the second work of this thesis, the proposed evolutionary approach extracts and selects such feature to build a parsimonious classifier. There are, however, other types of decision making problems, e.g., designing of an engine where the solutions need to be robust with respect to perturbations in their parameters. Often such a problem is formulated as a multiobjective fuzzy group decision making problem. In this context, we prefer solutions that are robust with respect to perturbations in the variable space as well as to their degree of consensus. We address this issue in our third work, where we propose a new measure called robust consensus and incorporate it in the multiobjective evolutionary search process. In the last work of this thesis, we propose a measure of sensitivity which can assess the sensitivity of a solution with respect to perturbations in the parameter space and use the same in the evolutionary search process. Here, we propose three approaches to reformulate a multiobjective optimization problem (MOP). The first approach provides a set of solutions with varying degree of sensi-

tivity, whereas, the other two approaches provide solutions with sensitivities below a given threshold. In this chapter, next we introduce some concepts that are required to present the works in this thesis.

1.1.1 Evolutionary Approaches: A Brief Introduction

Computational Intelligence (CI) deals with biologically and linguistically inspired computing paradigms. Evolutionary computation (EC) is one of the major components of CI. *Evolutionary algorithms* (EAs) exploit *Darwinian principles* to find solutions to a problem. These are, indeed, trial and error based optimization schemes that use metaheuristics. Moreover, EAs use a population consisting of a set of candidate solutions instead of iterating over a single solution in the search space. Usually, the following four techniques are categorized as EC: (i) *evolutionary programming* (EP), (ii) *evolutionary strategy* (ES), (iii) *genetic algorithm* (GA), and (iv) *genetic programming* (GP).

EC usually initializes a population with a set of randomly generated candidate solutions. However, if domain knowledge is available, it can be used to generate the initial population and this may significantly speed up the convergence to a desired solution. Then, the population is *evolved*. This evolutionary process incorporates *natural selection* and other *evolutionary* operators. From an algorithmic point of view, it is a guided random search that uses parallel processing to achieve the desired solutions. Note that, the *natural selection* must be incorporated in an EA, otherwise the approach cannot be categorized as an EC technique. For example, though several metaheuristic algorithms, such as, particle swarm optimization (PSO) [6] and ant colony optimization (ACO) [7, 8] are *nature inspired algorithms* (NIAs), they are not EAs. Note that, sometimes they are still loosely referred to as EC techniques.

In 1948, in a technical report [9], titled "Intelligent Machinery", written for National Physics Laboratory, Alan M. Turing wrote, "There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being survival value. The remarkable success of this search confirms to some extent the idea that intellectual activity consists mainly of various kinds of search." To the best of our knowledge, this is the first technical article, where the concept of evolutionary computation is found. However, it took few more decades to develop the following three distinct interpretations of this philosophy: (i) EP, (ii) ES, and (iii) GA. For the next one and half decades, these three areas grew separately. Later, in the early nineties, they were unified as a subfield of the same technology, namely EC. Each of EP, ES, and GA is an algorithm for finding solutions of an optimization problem - it finds a parameter vector that optimizes an objective function or a set of objective functions. Unlike

these three branches, GP finds a program to solve a problem. The concept of modern *tree-based* GP was proposed by Cramer in 1985 [10]. Later, Koza, a student of Cramer, popularized it with his many eminent works [11–14].

Simultaneous optimization of multiple conflicting objectives is frequent in real world problem solving. This type of optimization problems are called multiobjective optimization problems (MOPs). Consequently, researchers started solving MOPs using evolutionary approaches. In 1967, a Ph.D. thesis [15] hinted the use of evolutionary approaches to solve an MOP [16]. In [15], however, the author did not use any multiobjective evolutionary approach [16]. Instead, the author [15] restated the MOP as a single objective problem, and then, solved it using a genetic algorithm [16]. Though there is another rarely mentioned attempt in [17] to solve an MOP using a multiobjective evolutionary algorithm (MOEA), vector evaluated genetic algorithm (VEGA) [18] is usually considered to be the first popular evolutionary approach to solve an MOP [16]. In this thesis, we use evolutionary multiobjective approaches, and hence, in the next subsection we formally introduce MOPs.

1.1.2 Multiobjective Optimization

Without any loss of generality, a constrained MOP can be defined as

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathcal{V}}{\text{minimize}}\ \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x}));\\
&\text{subject to}\\
&g_j(\mathbf{x}) \leq 0, j = 1, 2, \cdots, n_{\neq};\\
&h_k(\mathbf{x}) = 0, k = 1, 2, \cdots, n_{=};
\end{aligned} \tag{1.1.1}$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$, $\mathbf{f} : \mathcal{V} \rightarrow \mathcal{O}$, n is the number of variables, m is the number of objectives, $n_{\neq}$ is the number of inequality constraints, and $n_{=}$ is the number of equality constraints. Here, $\mathcal{V}$ $(\subseteq \mathbb{R}^n)$ and $\mathcal{O}$ $(\subseteq \mathbb{R}^m)$ are respectively the feasible space and the objective space. A solution $\mathbf{x}_1 \in \mathcal{V}$ is said to dominate a solution $\mathbf{x}_2 \in \mathcal{V}$ if $\forall i\ f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2)$ and $\exists j$, such that, $f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2)$. It is denoted as $\mathbf{x}_1 \preceq \mathbf{x}_2$. Moreover, the overall constraint violation $(\mathcal{CV}(\cdot))$ of a solution $\mathbf{x} \in \mathcal{V}$ is defined as follows.

$$\mathcal{CV}(\mathbf{x}) = \sum_{j=1}^{n_{\neq}} < g_j(\mathbf{x}) > + \sum_{k=1}^{n_{=}} |h_k(\mathbf{x})|, \tag{1.1.2}$$

where

$$< z >= \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{1.1.3}$$

If $\mathbf{x}$ satisfies all the constraints, i.e., $\mathcal{CV}(\mathbf{x}) = 0$, $\mathbf{x}$ is said to be a feasible solution. Otherwise, it is an infeasible solution. Moreover, $\mathbf{x}_1$ is said to constraint dominate $\mathbf{x}_2$, denoted by $\mathbf{x}_1 \preceq_c \mathbf{x}_2$, if either (i) $\mathcal{CV}(\mathbf{x}_1) = 0$, $\mathcal{CV}(\mathbf{x}_2) = 0$, and $\mathbf{x}_1 \preceq \mathbf{x}_2$; or (ii) $\mathcal{CV}(\mathbf{x}_1) < \mathcal{CV}(\mathbf{x}_2)$. A solution $\mathbf{x}^{\odot} \in \mathcal{V}$ is called a Pareto optimal solution if $\nexists \mathbf{x} \in \mathcal{V}$, such that, $\mathbf{x} \preceq_c \mathbf{x}^{\odot}$. The set $PS = \{\mathbf{x}^{\odot} | (\nexists \mathbf{x} \in \mathcal{V}) \wedge (\mathbf{x} \preceq_c \mathbf{x}^{\odot})\}$ is called the Pareto set (PS), and the set $PF = \{\mathbf{f}(\mathbf{x}^{\odot}) | \mathbf{x}^{\odot} \in PS\}$ is called the Pareto front (PF). Due to the conflicting nature of the objectives, the aim of solving an MOP is to find a set of non-dominated solutions with the following two properties. First, a set of solutions should be as close as possible to the PF. Second, it should be uniformly distributed over the entire PF.

Since in this thesis we solve some pattern recognition problems using multiobjective evolutionary approaches, next we briefly introduce some necessary concepts of pattern recognition.

1.1.3 Pattern Recognition

Pattern recognition, a branch of machine learning (ML), focuses on the recognition of patterns and regularities in data. In this subsection, we discuss three tasks related to pattern recognition: classification, feature selection, and feature extraction, in brief.

1.1.3.1 Classification

Classification is one of the most important and frequently encountered problems in data mining and machine learning. A wide range of real world problems of different domains can be restated as classification problems. These include diagnosis from microarray data, text categorization, medical diagnosis, software quality assurance, and many more. Classification can be formally defined as follows. Let there be a given (training) data set $\mathcal{D} = \{(\mathbf{x}_k, y_k) : k = 1, 2, \cdots, n\}$, where $\mathbf{x}_k = (x_{k1}, x_{k2}, \cdots, x_{kd})^T \in F_S \subseteq \mathbb{R}^d$ is the k^{th} training pattern and $y_k \in Y = \{\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_c\}$ is the class label of $\mathbf{x}_k$. Here, the vector $\mathbf{f}_j = (x_{1j}, x_{2j}, \cdots, x_{nj})^T \in \mathbb{R}^n$ is called the j^{th} feature vector and F_S is called feature space, where, n, d, and c respectively denote the number of samples in the training data, the dimension of the feature space, and the number of classes.

Then, the *classification* task can be stated as learning of a mapping or a model, called a *classifier*, $\mathfrak{C} : F_S \rightarrow Y$, such that, given an unknown pattern $\mathbf{x} \in F_S$, $\mathfrak{C}(\mathbf{x}) = y_p \in Y$ be the correct class label associated with $\mathbf{x}$. The learning process derives a set of criteria from $\mathcal{D}$, and then, encodes them in $\mathfrak{C}(\cdot)$. High values of d, c, and feature-to-sample-ratio (d/n) usually increase the difficulties of classification. Redundant and/or noisy features also cause difficulties in classification. Sometimes, an ensemble of classifiers is used to obtain a better classification accuracy.

1.1.3.2 Feature Selection

Any classifier is designed using a set of features. Every feature may not be equally effective in designing a classifier. Some features may be derogatory, while some others can be non-informative for the classification task. Hence, *feature selection* (FS) becomes an important part in classification. For a given classification task, there may be at least four types of features: (i) essential, (ii) bad, (iii) redundant, and (iv) indifferent features [19]. The objective of a FS scheme should be to (i) select the essential features, (ii) reject the bad features, (iii) judiciously select some of the redundant features, and (iv) reject the indifferent features. Let us consider a small example to illustrate these four types of features [19]. Consider a data set on humans with five features: (i) sex, (ii) eye color, (iii) height, (iv) weight, and (v) number of legs. Suppose the given classification task has the following four classes:
(i) male AND (heavy weight OR long height),
(ii) male AND (low weight OR short height),
(iii) female AND (heavy weight OR long height), and
(iv) female AND (low weight OR short height).
In this particular case, (i) the feature *sex* is essential, (ii) the feature *eye color* is bad as it may confuse the learning, (iii) one of the features, *height* and *weight*, is redundant as a long person is usually heavy. Thus, *weight* and *height* constitute a redundant set for the given classification task because *usually* any one of the two will be enough for the classification task. Note that, we have emphasized on the word *usually* because height and weight are statistically strongly correlated, but there could be a heavy person with a short height. (iv) Finally, *number of legs* is an indifferent feature as under normal circumstances, it is going to be two for every individual. Note that, keeping some level of redundancy in the selected set of features may sometimes be helpful. Therefore, in the given context, one may argue that to account for some measurement error, it may not be a very bad idea to use both *height* and *weight* to design a classifier because that would be more robust than a classifier designed using only one of these two features.

FS can be formally defined as follows. Let there be a classification task associated with a training data set $\mathcal{D}$ with the set of features $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_d\}$ and the class label vector $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$. Then, FS is a process of judiciously selecting a subset of features $\mathcal{F}_\mathcal{R} \subseteq \mathcal{F}$, such that, if a classifier $\mathfrak{C}_S(\cdot)$ is trained using the reduced feature set $\mathcal{F}_R$, the performance of $\mathfrak{C}_S(\cdot)$ should be satisfactory. A good FS mechanism improves the associated classifier in the following manner: (i) FS simplifies the models to enhance their interpretability, (ii) FS reduces the degrees of freedom of the classifier, which in turn enhances the generalization capability of the classifier, and (iii) the training and decision making time may be reduced by a noticeable amount. Sometimes, FS also improves the performance of the classifier. Note that, FS could be unsupervised also, i.e., without using class label information.

FS methods are generally divided into two main groups: *filter* method and *wrapper* method [20]. A filter method does not use any feedback from the classifier or any mining algorithm. It relies on the general characteristics of data. On the contrary, to measure the goodness of features, a wrapper method uses a predetermined classifier or mining algorithm, which finally use the selected features. Consequently, a wrapper method exploits interactions among the selected features on which it is tested. But, to find an optimal set of features, a wrapper method needs to measure performances on all possible subsets of features. This becomes infeasible for high dimensional data sets. To overcome this problem, wrapper methods typically use a heuristic-based forward or backward selection mechanism [20], which does not evaluate all possible subsets. There is a third category of methods, called *embedded* methods, where FS and designing of the classification system are done *together* in an integrated manner. Embedded methods do not need to evaluate all possible subsets. Moreover, they can account for interaction between features and the classifier that is used to solve the problem [21]. Usually, embedded methods attain comparable accuracy to wrapper methods as well as comparable efficiency to filter methods. Though for every classification tool it may not be easy to define such an integrated mechanism, several attempts of FS using embedded methods have already been made. These attempts include using single objective GP [22], neural networks [19, 23], and support vector machines [24]. In the literature, there are also some methods [25–28] for FS based on Markov blanket [29]. Among these works, in [26] and [27] embedded methods of FS using single objective and multiobjective genetic algorithms have been proposed, respectively. These works minimize the overall error or the class-specific errors using one-vs.-all strategy, and make use of the Markov blanket property. If a feature has a Markov blanket within the selected set of features, then that feature does not provide any more information

beyond the Markov blanket about the class labels and other selected features [27], and hence, can be discarded.

1.1.3.3 Feature Extraction

For a given problem, it may happen that the input data dimension is too large to be processed, the feature set has redundancy, and may even lack relevant features. Then a process, called *feature extraction* (FE), is used to construct new features. Usually both FS and FE are considered to be different ways of achieving dimensionality reduction. However, to enhance separability, sometimes FE can be used to increase the dimensionality of the features. To illustrate this with an example, let us consider a XOR-type, binary class data set $\mathcal{D}_{\oplus} = \{((0,0), y_-)$, $((0,1), y_+)$, $((1,0), y_+)$, $((1,1), y_-)\}$ with $Y_{\oplus} = \{y_-, y_+\}$. The two classes are not linearly separable. But, if we concatenate an extracted feature $f_E = x_1 \times x_2$, and convert this data set as $\mathcal{D}_{\oplus}^{f_E}$=$\{((0,0,0), y_-)$, $((0,1,0), y_+)$, $((1,0,0), y_+)$, $((1,1,1), y_-)\}$, then the two classes become linearly separable. We can formally define FE as follows. Let there be a classification task associated with a training data set $\mathcal{D}$. Then, FE is the process of finding a mapping $\mathfrak{F}_E : F_S \rightarrow F_S^E$, such that, $F_S^E \subseteq \mathbb{R}^{d_E}$, and if a classifier $\mathfrak{C}_E(\cdot)$ is trained using $\mathcal{D}_E = \{(\mathfrak{F}_E(\mathbf{x}_k), y_k) : k = 1, 2, \cdots, n\}$, the performance of $\mathfrak{C}_E(\cdot)$ should be satisfactory. For a classification problem, the ultimate goal of FE is to enhance the classifier performance. So for a FE algorithm my use classifier performance as the objective. But other FE algorithms may use other criteria such as preservation of interpoint distances or autoencoding error. While FS enhances classification performance by selecting a smaller subset of features, FE constructs new features to achieve the same.

Let us revisit the example on the human data set. If we want to employ FE in that scenario, it would be good to construct a feature *height 'OR' weight*. Note that, this *'OR'* operator is not exactly the Boolean OR operator because the attributes heavy, long etc. are not Boolean, but fuzzy concepts, and hence, such a combined feature has to be designed judiciously. The beauty of a GP-based system is that, during the evolution, it may compute the intuitive 'OR' operator using the members of $\mathcal{F}$.

In the next section, we will discuss how GP can be used for pattern classification, FS, and FE. In this context, we will also discuss some state-of-the-art methodologies that employs GP for these pattern recognition tasks.

1.1.4 Genetic Programming in Pattern Recognition

GP [11–13, 30, 31] finds computer programs to solve a given problem. It evolves the programs using biologically inspired genetic operations like reproduction, crossover, and mutation in the search space [32–34]. GP consists of a set of instructions and a fitness function to evaluate the performance of a candidate computer program. Therefore, it can be considered a special case of GA, where each solution is a computer program. There exist several ways to encode a program. We use a tree structure [10, 35–37], which is probably the most popular and traditional one. In a tree-based GP, the internal nodes of the trees must be from a set of predefined functions (operators), $\mathcal{G}$. Moreover, every leaf node of a tree must be from a set of predefined terminals (operands), $\mathcal{T}$. The subtrees of a function node $g \in \mathcal{G}$ are the arguments to g. Note that, a very important property of tree-based GP is that, $\mathcal{G}$ and $\mathcal{T}$ need to satisfy both the *closure* property and the *sufficiency* property [11, 34]. To satisfy the closure property, $\mathcal{G}$ needs to be well defined and closed for any combination of probable arguments that it may encounter [11, 34]. Moreover, $\mathcal{G}$ and $\mathcal{T}$ need to be able to encode any possible valid solution of the problem to satisfy the sufficiency property. Thus, Lisp or any other functional programming language that naturally embodies tree structures, can be used to represent a candidate solution in GP. Use of non-tree representations to encode solutions in GP, is comparatively less popular. An example of this is linear genetic programming, which is suitable for more traditional imperative languages. In this thesis, we use only on *tree-based* GP. The most frequently used representations of tree-based GPs are decision trees, classification rules, and discriminant functions (DFs). In this thesis, we confine our discussion primarily to DF based GP.

Since, in this thesis, we have used multiobjective genetic programming (MOGP) for learning DFs, we discuss some DF based systems. In a DF based GP, each program is a mathematical equation, where the variables are features of the data. Usually every program is associated with a class. For every input data point, the program converts it to a single output value. If this output value is more than a predefined threshold (usually zero), the point belongs to the class to which the program is associated with. Thus, a single equation is enough for binary classification problems. For c-class problems, there are two common ways. The first and more frequently used approach is to consider a c-class problem as c binary classification problems [38]. Thus, c number of DFs are used to discriminate c classes. The second and less popular approach is to use a single DF with $(c - 1)$ threshold values to create c intervals. Each of these intervals is then assigned to a particular class. For both categories, a common practice is to evolve a population where each individual encodes either one (for a binary

classification problem having a single threshold, or for a multi-class problem having multiple thresholds) or multiple DFs each of which uses a single threshold. Such encoding schemes have been used in [22,34]. In another approach, each solution encodes multiple DFs having a single threshold. The final output value, in this case, may be obtained using a (weighted) voting scheme among the output values of different functions of the same individual. An example of this scheme can be found in [39].

Many authors [40–47] have used GP to design classifiers or to generate rules for binary classification problems. Some researchers have also attempted to solve multi-class problems [34, 48–51]. GP has also been used for FS [52], FE [53], and simultaneous FS and classification [22]. In [39], the authors have used GP to create ensembles of classifiers (genetic programs). They have used these ensembles to classify microarray data sets. Though GP is a powerful tool, it has a drawback: without a special care each genetic program (equation) becomes huge. As an effect, they may not learn the patterns in the training data, rather, memorize them. It also makes genetic programs difficult to comprehend. Besides, though ensembles can perform better than individual classifiers [54], to obtain better performance, each ensemble should be diverse and each member of the ensemble should be accurate [54–56]. Due to the lack of an explicit diversity preservation mechanism, without special care, the solutions of a single objective GP may loose diversity. Note that, a genetic program may not (mostly will not) use all features of a given data set. Hence, a GP-based system performs FS implicitly, at least to some extent even if it is not specially designed for FS. Moreover, a DF based genetic program also implicitly performs FE from an initial set of measured features. The derived features are expected to be less-redundant, informative, and should facilitate subsequent learning and enhance generalization. Sometimes, they may lead to a better human interpretation. In Chapters 2 and 3, we have provided detailed literature surveys related to GP in classification, FS, and FE.

1.1.5 Robust Optimization

Up to this point, we have introduced evolutionary approaches, MOPs, three pattern recognition tasks, and the use of GP in these pattern recognition tasks, which cover the necessary prerequisites of Chapters 2 and 3. In this subsection, we discuss the preliminaries and a brief literature review on robust optimization, which are related to our works presented in Chapters 4 and 5.

Uncertainties in real world MOPs are sometimes unavoidable. Primarily, there can be three types of uncertainties [57–59]: 1) uncertainties in the environment and operating conditions, 2) uncertainties in the parameters, and 3) uncertainties in gen-

erating the output. The first two types of uncertainties are particularly important in optimizing real-world problems [57]. A typical example of uncertainty would be the measurement noise in the values of the variables during the evaluation of a given solution. The evaluating function may even be noisy [60]. Existence of such noise may mislead the search process [60–62]. Optimization with uncertainties is dealt within a framework called robust optimization [60–64].

The literature on robust optimization in MOP is quite rich [57, 60, 62–72]. In this subsection, we discuss a few state-of-the-art works that are the most relevant to our investigation. In one of the initial major investigations on robustness in multiobjective optimization, Deb and Gupta [63] defined robust solutions using both expectation-based and variance-based approaches. The authors [63] respectively denoted them as robust solutions of type I and type II.

If a solution $\mathbf{x} \in \mathcal{V}$ is in the PS of the following multiobjective minimization problem, it is called a multiobjective robust solution of type I:

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathcal{V}}{\text{minimize}}\, \mathbf{f}^e(\mathbf{x}) = (f_1^e(\mathbf{x}), f_2^e(\mathbf{x}), \cdots, f_m^e(\mathbf{x}));\\
&\text{subject to}\\
&g_j(\mathbf{x}) \leq 0, j = 1, 2, \cdots, n_{\neq};\\
&h_k(\mathbf{x}) = 0, k = 1, 2, \cdots, n_{=};
\end{aligned} \tag{1.1.4}$$

where $f_j^e(\mathbf{x})$ is called the mean effective objective function and it is defined as

$$f_j^e(\mathbf{x}) = \frac{1}{|\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})|} \int_{\mathbf{y}\in\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})} f_j(\mathbf{y}) d\mathbf{y}. \tag{1.1.5}$$

Here, $\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})$ denotes a small neighbourhood of $\mathbf{x}$ in the variable space defined with the help of parameter δ, and $|(\cdot)|$ indicates the hypervolume of $(\cdot)$.

If a solution $\mathbf{x} \in \mathcal{V}$ is in the PS of the following multiobjective minimization prob-

lem, it is called a multiobjective robust solution of type II:

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathcal{V}}{\text{minimize}}\, \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x}));\\
&\text{subject to}\\
&g_j(\mathbf{x}) \leq 0, j = 1, 2, \cdots, n_{\neq};\\
&h_k(\mathbf{x}) = 0, k = 1, 2, \cdots, n_{=};\\
&\frac{||\mathbf{f}^e(\mathbf{x}) - \mathbf{f}(\mathbf{x})||_p}{||\mathbf{f}(\mathbf{x})||_p} \leq \eta;
\end{aligned} \tag{1.1.6}$$

where η is a limiting parameter, which is usually constant throughout the optimization process. Note that, though the definition uses p-norm ($||\cdot||_p$), usually $p = 2$, i.e., the Euclidean norm, is used.

When we deal with robust solutions of type I and type II, we need to incur a high cost to attain a satisfactory approximation of each mean effective objective function. In [73] a method of surrogate modelling has been proposed to reduce this cost for single objective robust optimization. The surrogate model in [73] uses radial basis functions for approximation. A new evolutionary algorithm, which uses max-min optimization strategy, has also been proposed in [73].

Bui et al. [60] defined two types of robustness: dominance robustness (DR) and preference robustness (PR). DR is defined as the ability of a Pareto optimal solution to stay in the PF when it is perturbed in the variable space. For a non-dominated solution $\mathbf{x} \in \mathcal{V}$, DR is quantified as follows.

$$\text{DR}(\mathbf{x}) = \frac{1}{|\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})|} \int_{\mathbf{y}\in\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})} \mathcal{G}(\mathbf{y})\, d\mathbf{y}, \tag{1.1.7}$$

where $\mathcal{G} : \mathbb{R}^n \to \mathbb{R}$ is a "dominance function" [60], such that, if $\mathbf{y}$ is a non-dominated solution $\mathcal{G}(\mathbf{y}) = 0$, else $\mathcal{G}(\mathbf{y}) = 1$. If $\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})$ is a countable set, $\text{DR}(\mathbf{x})$ is defined as follows.

$$\text{DR}(\mathbf{x}) = \frac{1}{|\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})|} \sum_{\mathbf{y}\in\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})} \mathcal{G}(\mathbf{y}). \tag{1.1.8}$$

On the contrary, for a given Pareto optimal solution, PR is defined as the minimum transition cost in the variable space when the Pareto-optimal solution is perturbed in

the objective space. For a non dominated solution $\mathbf{x} \in \mathcal{V}$, PR is quantified as follows.

$$\mathrm{PR}(\mathbf{x}) = \frac{1}{|\mathcal{B}_\delta^{\mathcal{O}}(\mathbf{f}(\mathbf{x}))|} \int_{\mathbf{f}(\mathbf{y}) \in \mathcal{B}_\delta^{\mathcal{O}}(\mathbf{f}(\mathbf{x})) \bigcap \mathcal{F}_\mathcal{P}} c\left(\mathbf{f}\left(\mathbf{y}\right)\right) d\mathbf{f}\left(\mathbf{y}\right), \tag{1.1.9}$$

where $\mathcal{B}_\delta^{\mathcal{O}}(\mathbf{f}(\mathbf{x}))$ denotes a small neighbourhood of $\mathbf{f}(\mathbf{x})$ in the objective space defined with the help of parameter δ, $\mathcal{F}_\mathcal{P}$ denotes the PF, and $c\left(\mathbf{f}\left(\mathbf{x}\right)\right) : \mathbf{f}\left(\mathbf{x}\right) \rightarrow \mathbb{R}$, where $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$ is an expected cost function. $c(\cdot)$ "quantifies the cost incurred in decision space when $\mathbf{f}(\mathbf{x})$ is moved to a neighboring point in the m-dimensional objective space" [60]. Depending on the problem domain, the transition cost may change. In general, $c(\cdot)$ can be described as the financial loss for changing from a solution to another. Alternatively, it can be the additional cost to generate a new solution. If there is a finite set of solutions in the neighbourhood, and for simplicity, if the cost is considered the average Euclidean distance between $\mathbf{x}$ and its neighbors, $\mathrm{PR}(\mathbf{x})$ is defined as follows.

$$\mathrm{PR}(\mathbf{x}) = \frac{1}{|\mathcal{B}_\delta^{\mathcal{O}}(\mathbf{f}(\mathbf{x}))|} \sum_{\mathbf{f}(\mathbf{y}) \in \mathcal{B}_\delta^{\mathcal{O}}(\mathbf{f}(\mathbf{x})) \bigcap \mathcal{F}_\mathcal{P}} ||\mathbf{x} - \mathbf{y}||_2. \tag{1.1.10}$$

In Chapter 4 and 5, we have provided detailed literature surveys related to robust optimization in MOPs.

1.1.6 Fuzzy Group Decision Making in Multiobjective Optimization

As the last introductory topic, here we discuss fuzzy group decision making in the context of robust multiobjective optimization, which is required as a prerequisite for our work presented in Chapter 4.

If a set of decision makers (DMs) is involved in solving a problem, it is categorized as a group decision making (GDM) problem. When a GDM problem is also an MOP, it is called a group decision making for a multiobjective optimization problem (GDM-MOP) [74]. Sometimes, the DMs are weighted and the weights are provided by another *expert*. In a given GDM-MOP, if the DMs provide their preferences using fuzzy numbers, the problem is called fuzzy group decision making for a multiobjective optimization problem (FGDM-MOP) [74].

Now, we mathematically formulate an FGDM-MOP problem. Let there be d DMs and the weight vector associated with the DMs be $\mathbf{w} = (w_1, w_2, \cdots, w_d)^T$, such that, $w_i \in (0,1)$ and $\sum_{i=1}^{d} w_i = 1$. We also assume that the i^{th} DM provides her preference using either a (fuzzy) reference point for the j^{th} objective $\mathbf{r}_{ij}^{\mathcal{O}}; i = 1, 2, \cdots, d; j =$

$1, 2, \cdots, m$; or a (fuzzy) reference point $\mathbf{r}_{ij}^{\mathcal{V}}; i = 1, 2, \cdots, d; j = 1, 2, \cdots, n$; for the j^{th} variable. When the preference is given in the objective space, a DM may give her preference as CLOSE to $\mathbf{c} = (c_1, c_2, \cdots, c_m)^T \in \mathbb{R}^m$ or can express her preference as m different preferences as CLOSE to $c_i; i = 1, 2, \cdots, m$. Similarly, preferences in the variable space can be expressed as either CLOSE to $\mathbf{c} = (c_1, c_2, \cdots, c_n)^T \in \mathbb{R}^n$ or CLOSE to $c_i; i = 1, 2, \cdots, n$. Note that, to keep the notation simple, we are ignoring the superscript $\mathcal{V}$ or $\mathcal{O}$. We shall restrict ourselves to the case when preferences are given for each variable (or objective) separately. The extension to the multi-dimensional case is straightforward. The DM may explicitly define what she meant by CLOSE to. Typically, the concept CLOSE to is expressed by a triangular/trapezoidal/Gaussian membership function. The concept of CLOSE to, although usually is symmetric, asymmetric function may also be used depending on the problem and preference of the DM. We shall restrict ourselves to symmetric triangular or Gaussian memberships. Thus, to define each membership, we need two parameters, the *center* and the *width*. If the DMs do not provide the *width* parameters, we can assign a fixed *width* to each membership function. Thus, in the present case, $\mathbf{r}_{ij}^{\mathcal{O}} = \left(c_{ij}^{\mathcal{O}}, s_{ij}^{\mathcal{O}}\right)^T$ and $\mathbf{r}_{ij}^{\mathcal{V}} = \left(c_{ij}^{\mathcal{V}}, s_{ij}^{\mathcal{V}}\right)^T$. The associated membership functions are respectively denoted by $\mu_{ij}^{\mathcal{O}}(\cdot)$ and $\mu_{ij}^{\mathcal{V}}(\cdot)$. Two plausible definitions of CLOSE to c (with spread s) are Gaussian membership function $\mu_G(\cdot)$ and triangular membership function $\mu_T(\cdot)$, defined as follows:

$$\mu_G(z; c, s) = e^{-\frac{(z-c)^2}{2s^2}}; \tag{1.1.11}$$

and

$$\mu_T(z; c, s) = \begin{cases} 1 - |z - c|/s, \text{ if } (c - s) \leq z \leq (c + s) \\ 0, \text{ otherwise.} \end{cases} \tag{1.1.12}$$

In Chapter 4, we have provided a literature survey related to FGDM-MOPs.

1.2 The Scope of the Thesis

As mentioned earlier, in this thesis, we have addressed four research problems related to decision making using multiobjective evolutionary approaches. The first two of them are related to pattern classification, which is usually the first step of decision making. There are other kinds of decision making problems, such as, designing of an

engine where the design parameters must be robust with respect to input perturbations. The third problem that we deal with in this thesis is related to robust optimization. The last problem that we deal with is another kind of decision making, where multiple experts are involved in the decision making process. Here, each experts has a preference about the solution. Our objective is to obtain a set of solutions that is robust with respect to input perturbations as well as the set enjoys a high degree of consensus. Below, we shortly describe the primary contributions in the chapters to follow.

1.2.1 A Multiobjective Genetic Programming based Ensemble for Simultaneous Feature Selection and Classification [1]

In Chapter 2, we present an integrated algorithm for simultaneous FS and designing of diverse classifiers using a steady state MOGP, which minimizes three conflicting objectives: false positives, false negatives, and the number of leaf nodes in the tree. Our method divides a c-class problem into c binary classification problems. It evolves c sets of genetic programs to create c ensembles. During the mutation operation, our method exploits the fitness as well as the unfitness of features, which dynamically change with generations with a view to using a set of highly relevant features with low redundancy. The classifiers of the i^{th} class determine the *net belongingness* of an unknown data point to the i^{th} class using a weighted voting scheme, which makes use of the false positive and false negative mistakes made on the training data. We test our method on eight microarray and eleven text data sets with a diverse number of classes (from 2 to 44), a large number of features (from 2000 to 49151), and a high feature-to-sample ratio (from 1.03 to 273.1). We compare our method with a bi-objective genetic programming scheme that does not use any FS and rule size reduction strategy. It helps to depict the effectiveness of the proposed FS and rule size reduction schemes. Furthermore, we compare our method with four classification methods in conjunction with six features selection algorithms and the full feature set. Our scheme performs the best for 380 out of 474 combinations of data sets, classification algorithm and FS method.

1.2.2 Feature Extraction and Selection for Parsimonious Classifiers with Multiobjective Genetic Programming [2]

The primary objective of the work in Chapter 3 is to design classifiers that are reasonably simple and understandable yet capable of yielding a good performance. Thus,

our problem here is to obtain an ensemble of linear classifiers with as few simple features as possible. For this we use GP to extract features that can linearly separate two classes. We propose an integrated mechanism for simultaneously extracting and selecting useful linearly separable features. We decompose a c-class problem into c binary classification problems and evolve c sets of binary classifiers employing a steady-state MOGP with three minimizing objectives. Each binary classifier is composed of a binary tree and a linear support vector machine (SVM). The features extracted by the feature nodes and some of the function nodes of the tree are used to train the SVM. The decision made by the SVM is considered the decision of the corresponding classifier. During crossover and mutation, the SVM-weights are used to determine the usefulness of the corresponding nodes. We also employ unfitness functions for the feature nodes and a fitness function based on Golub's index to select useful features. We compare our method with 34 classification systems using 18 data sets. The performance of the proposed method is found to be better than 432 out of 570, i.e., 75.79% of comparing cases. Our results confirm that the proposed method is capable of achieving our objectives.

1.2.3 Robust Multiobjective Optimization with Robust Consensus [3, 4]

In Chapter 4, we consider a multiobjective robust optimization problem where a set of weighted DMs provides their preferences a priori. The preferences are provided either in the objective space or in the decision variable space using fuzzy numbers. To solve this problem, the following three things are required: 1) an indicator to measure consensus, 2) an indicator to measure the robustness of the solutions to their degrees of consensus, and 3) a reformulation of the multiobjective robust optimization problem. It is necessary for the reformulated problem to generate robust solutions that also enjoy high degrees of consensus. In this work, we have addressed these three issues. For this purpose, we have proposed two approaches to define *consensus*. Then, we have extended these approaches to define *robust consensus*, an indicator to measure the robustness of a given solution to its degree of consensus. Though these approaches can be used to define a countless number of measures, we have proposed 12 definitions of consensus, and hence, robust consensus. Furthermore, we have proposed two ways for the reformulation. Experimental results illustrate that the behaviors of the proposed definitions and of the reformulations are consistent with our expectations.

1.2.4 Robustness in Multiobjective Evolutionary Optimization [5]

In Chapter 5, in the context of multiobjective optimization problems, we have introduced a new measure of sensitivity, which is inversely related to robustness of solutions. It quantifies the robustness of a solution with respect to perturbations in the variable space. We have shown how the cost of computing this measure can be reduced using an approximation with the first-order Taylor series expansion subject to the following three conditions. First, the objective functions are differentiable. Second, the input noise is any one of uniform noise, additive white Gaussian noise, and multiplicative noise. Third, the changes in the variables are very small. Next, we have proposed three approaches to reformulate MOPs. When the first approach is used, solving the reformulated MOP yields solutions of the original MOP with different degrees of sensitivity/robustness. When the other two approaches are used, the reformulated MOPs yield solutions with sensitivities less than a given threshold. We have experimentally validated our claims and have analyzed the behaviour of the reformulation strategies.

1.2.5 Conclusions, Limitations, and Future Scopes

We conclude the thesis in Chapter 6. In this Chapter, we discuss some limitations of the work presented in this thesis along with probable ways of extending our work in future.

Chapter 2

A Multiobjective Genetic Programming based Ensemble for Simultaneous Feature Selection and Classification [1]

2.1 Introduction

The microarray technology has made it possible to diagnose different types of cancers directly using the microarray data sets. Again, finding keywords as well as contexts from text data is essential to detect (without human intervention) the context of web pages, emails, or questions/answers, etc. Consequently, it is an important task to perform feature selection (FS) and classification on both microarray and text data sets. However, the nature of these two data sets are quite distinct, and hence, they pose different challenges. Specifically, microarray data sets usually have a large number of features and a very small number of samples. This causes a very high feature-to-sample ratio. On the contrary, text data sets usually have a large number of classes, a large number of features, a large number of samples, and sparsity. In the literature, several attempts have been made to address the above issues, yet there is a need for finding better solutions.

The objective of this work is to find an embedded methodology of simultaneous FS and classification employing genetic programming (GP) as a tool, such that, it should perform satisfactorily when applied to both microarray and text data sets. Some of the

contributions of our schemes are as follows. We have introduced a new multiobjective genetic programming (MOGP), called archive-based steady-state micro genetic programming (ASMiGP). It is enriched by several new operators. The mating selection judiciously uses Roulette wheel selection instead of a two tire multiobjective selection scheme (where domination is preferred over diversity). The crossover is new and uses male-female differentiation so that the offspring is more likely to be close to the female parent in the genotypic space. The mutation is restrictive and performs less exploration in the hypothesis space. Thus, it reduces disruption. For feature nodes, instead of fitness, mutation uses unfitness to select the mutation point. Altering the fitness and unfitness of the features in different stages of learning, we change the objective of searching in the corresponding stages. For a c class problem, we use ASMiGP to learn c diverse sets of classifiers (equations) minimizing three objectives: false positive (FP), false negative (FN), and number of leaf nodes of a tree to restrict the rule size. Throughout the learning process, implicit FS is performed using MOGP, whereas several filter based approaches are used in different stages of the procedure. In this way, we obtain concise rules that involve only simple arithmetic operations. A weighted negative voting scheme is then used among the rules of each ensemble. These weights are determined on the basis of the performance of the binary classifiers on the training data set. A new measure of weighted negative voting, called *net belongingness*, is also introduced.

The proposed method has been tested on eight multi-class microarray data sets each having a large number of features, varying from 2000 to 49151, and a high feature-to-sample ratio, varying from 32.26 to 273.06. It has also been tested on eleven high dimensional (varying from 3182 to 26832) text data sets, where the number of classes (categories) vary from 6 to 44. Experimental results reveal that our method can generate ensembles of classifiers with concise rules that can do a good job of classification with a small subset of features.

2.2 Related Works

Many researchers have used GP as a tool for classification and FS. Some literature surveys on this topic can be found in [38,75]. It has been used in both filter [76–79] and wrapper [80–83] based approaches. GP has also been used for extracting decision trees [84–89]. Among these, in [87, 88], MOGP has been used. GP has also been adopted for learning rule based systems [41–43, 47, 90–92]. Both binary classification problems [40,41,44–47] and multi-class classification problems [34,42,43,48–51,90–92] have been

solved by GP based (rule based) systems. Even, researchers have applied GP based (rule based) system for binary classification of imbalanced data [93, 94]. Discriminant functions (DFs) based GP for online FS and classification has been adopted in [22] to solve multi-class problems. It is noteworthy that GP has also been used in feature extraction for edge detection [95].

In the recent years, researchers have made some notable attempts to solve classification problems using MOGP. In [96], the authors have proposed a MOGP to obtain a group of classifiers, with which the maximum receiver operating characteristic convex hull (ROCCH) can be obtained. For this purpose, they have adopted four different multiobjective frameworks into GP. For further improvement of each individual genetic program's performance, they have defined two local search strategies that have been especially designed for classification problems. The experimental results in [96] have demonstrated the efficacy of the proposed memetic scheme in their MOGP framework. Another convex hull-based MOGP, called CH-MOGP, can be found in [97]. In [97], the authors have shown the differences between the conventional multiobjective optimization problem (MOP) and ROCCH maximization problem. They [97] have introduced a convex hull based sorting without redundancy and a new selection procedure which are suitable for ROCCH maximization problem. In [56], the authors have proposed a MOGP based approach that is especially designed for imbalanced data sets. This approach [56] evolves ensembles of diverse and accurate GP classifiers with good performance on both the majority and the minority classes. The individual members of the evolved ensembles, that are composed of nondominated solutions, vote on class membership.

During evolution of GP, the variable length genome gradually starts to increase its length without significant improvement in fitness. This incident, called *bloating* [98], is a well known phenomenon in GP. Bloating causes genetic programs (i) to keep reducible genome structures, and (ii) to memorize training data points, rather than recognizing hidden patterns. To find rules, which are to some extent human interpretable and can be analyzed, each of the genetic programs must be concise. A plausible way to achieve this target is to control bloating. A popular way to handle bloating is to take into account the program size [11, 99–102]. Some other methods include spatially-structured populations [103, 104], island based models to introduce spatial structure to the population [103, 104], intron deletion [105], and dynamic population sizing [106, 107]. The work in [108] attempts to understand the GP evolved solutions; while authors in [109] attempt to find comprehensible rules in subgroup discovery.

In classification, the basic task is to search through a hypothesis space to find a

suitable hypothesis that will be able to classify the data points in a better way. An ensemble combines multiple hypotheses to form a better one [39, 55, 56, 94, 110, 111]. Empirically, an ensemble performs better when each of the classifiers is highly accurate (strong learner) and the members of the ensemble are significantly diverse. The explanation behind the better performance of ensemble classifiers than a single classifier has been described in [54]. Normally, to decide the class label of a data point, the member classifiers of an ensemble use (weighted) voting although other aggregation schemes can and have been used. Ensembles are often used in bio-informatics classification problems [112].

2.3 Proposed Work

2.3.1 Representation of Classifiers or Solutions

Here, to solve a c class problem, we evolve c-populations of genetic programs. Each individual of these populations is a binary classifier. Each binary classifier is represented by a single tree. When a data point is passed through an individual of the i^{th} population, if the output value is positive, the individual says that the passed data point belongs to the i^{th} class; otherwise, it says that the point does not belong to that class. The internal (non-leaf) nodes of these trees are a set of functions $\mathcal{G}$. The terminal nodes must be either a feature or a constant from the set $\mathcal{C}$. We have imposed some constraints on the minimum size (architecture) of the trees. In each tree there must be at least one function node and two terminal nodes (with at least one feature node). Though, a single feature (terminal) node might be enough to determine the class label, this would rarely happen in practice. The above restrictions make the trees more useful without any loss of generalization capability. Again, after generation of any tree throughout the learning phase (using mutation, crossover, or random initialization), the largest subtree consisting of only constants as its leaf nodes is replaced by a constant leaf node having the equivalent constant value.

2.3.2 Multiobjective Genetic Programming for Learning

Larger genetic programs may memorize the training patterns which, in turn, may increase the training accuracy, and reduce the test accuracy and the understandability of the rules. Therefore, we aim to find smaller but accurate classifiers. Again, when c is high enough, even a balanced c-class data set may get converted to c number of highly imbalanced binary classification data sets. Instead of minimization of classifi-

Algorithm 2.1 Archive-based steady-state micro genetic programming (ASMiGP)

Initialize population using ramped-half-and-half method. Initialize the archive solutions using initial solutions. **while** $Evaluations^{Current} \leq Evaluations^{Maximum}$ **do**
  repeat
    operator = Select crossover or mutation. **if** *operator* == *crossover* **then**
      Select male and female parents (mating selection). Perform crossover.
    end
    else
      Select an individual (mating selection). Mutate the individual.
    end
  until *the infix equation of offspring is distinct from infix equation of any individual of the archive*
  Evaluate the new offspring. $Evaluations^{Current} = Evaluations^{Current} + 1$ Update the archive using new offspring (multiobjective environmental selection).
end
$\mathcal{F}ronts$ = Perform fast-non-dominated-sort. **return** first front of $\mathcal{F}ronts$.

cation error, simultaneous minimization of FP and FN would be more appropriate in this regard. Because in this case usually the size of the positive class is much smaller than that of the negative class. Consequently, minimization of the total misclassification error often may make the classifier essentially learn the negative class and practically reject the positive class (high FN). That is why we minimize both FP and FN. Thus, we have three objectives, i.e., minimizations of (a) FP, (b) FN, and (c) rule size. Moreover, when different classes are overlapped, minimization of FP is usually in conflict with the minimization of FN and vice versa. Multiobjective optimization is more suitable when we need to optimize more than one conflicting objective. Therefore, during the learning of each binary classifier, we minimize three objectives using an MOGP: (a) FP, (b) FN, and (c) the number of leaf nodes in the tree. The third objective is to reduce the size of the tree which enhances the understandability and reduces the pattern memorization capability. The algorithm, proposed in this work, is called ASMiGP: archive-based steady-state micro genetic programming, which is presented in Algorithm 2.1.

In an evolutionary search, it is desired to have as many generations as possible and steady state nature of an algorithm maximizes the number of generations when the number of function evaluations is fixed [113]. Due to maximization of generations, a steady state evolutionary search is more exploitative to enhance the searching in a region which is more likely to have or to be closer to the Pareto front and avoiding exploration in regions that are less likely to improve the solutions. It causes faster convergence. In other words, independent of the fitness evaluation process, steady

state selection performs better than discrete generational selection [114]. Therefore, we have used a steady state algorithm instead of a generational one.

2.3.3 Feature Selection

In this work, we have used the *embedded* model of FS. Explicit FS is performed during population initialization and mutation. Implicit FS is performed during crossover. *Filtering* is also performed at three different stages of the learning process; in particular, at the beginning and after 50% and after 75% of evaluations as described next in this subsection.

To facilitate the FS, following [115], we define an index that assesses the discriminating ability of a feature. Consider a two-class problem. Note that for a multi-class problem, the one-versus-all case can also be viewed as a two-class problem. Let there be n_p number of training points and the class label for the j^{th} data point be $+1$ if it belongs to class 1, and -1 if it belongs to class 2. Let the value of the f^{th} feature for the k^{th} data point be x_{fk}; $k = 1, 2, \cdots, n_p$. If the f^{th} feature is a good discriminatory feature then for all data points from class 1, it should have high values, and for all points from class 2, it should have low values or vice-versa. Hence for a good discriminatory feature, we can define an ideal feature vector with values 1, if the data point is from class 1 and 0 otherwise; or the feature value is 0, if it is from class 1, otherwise it is 1. Let y_{ik} be the ideal feature value of the k^{th} data point for the i^{th} classification task. Note that, there could be other important features that are not linearly related to the class structure. We are not considering them in this preliminary filtering step. As in [115], we compute the Pearson's correlation (or any other measure of similarity) between the ideal feature vector $(\mathbf{y}_i)$ and the f^{th} feature vector as a measure of feature relevance:

$$C_f^i = \frac{\sum_{k=1}^{n_p}(x_{fk} - \overline{x_f})(y_{ik} - \overline{y_i})}{\sqrt{\sum_{k=1}^{n_p}(x_{fk} - \overline{x_f})^2 \sum_{k=1}^{n_p}(y_{ik} - \overline{y_i})^2}}. \tag{2.3.1}$$

A higher value of $|C_f^i|$ indicates a stronger discriminative power of feature f.

Now, we describe the FS procedure. Let the set of all features be $\mathcal{F}_{all}$. We intend to incorporate only those features from $\mathcal{F}_{all}$ which are more likely to help the classifiers to decide the class label. We assign different fitness and unfitness measures to the features during the learning phase. To remove a feature node from any tree (during mutation), we select it using Roulette wheel selection on the unfitness values of the

features which are present in that tree. Similarly, when a new feature is inserted in the tree, it is selected using Roulette wheel selection on the fitness values. During the first 50% evaluations, the fitness of the features is defined as in (2.3.2), and the unfitness is defined as in (2.3.3).

$$F^{0\%,i}_{fitness}(f,i) = \begin{cases} \left(\dfrac{C^i_f}{C^i_{max}}\right)^2, & \text{if } \dfrac{\left|C^i_f\right|}{C^i_{max}} > 0.3 \\ 0, & \text{otherwise} \end{cases} \tag{2.3.2}$$

$$F^{0\%,i}_{unfitness}(f,i) = 1.0 - F^{0\%,i}_{fitness}, \tag{2.3.3}$$

where $C^i_{max} = \max_{f \in \mathcal{F}_{all}} \left|C^i_f\right|$. Equation (2.3.2) sets the fitness of very poor features (poor with respect to its discriminating power) to zero to eliminate their impact during the initial evolution. Let, $\mathcal{F}_{\text{eval}=0\%,i} \subseteq \mathcal{F}_{all}$ be the features with nonzero fitness values. Basically, at this stage we are using a filter on the feature set $\mathcal{F}_{all}$ to obtain a smaller feature set.

After completion of 50% evaluations for each population, we find the features used in that population. Let the feature set be $\mathcal{F}_{\text{eval}=50\%,i}$. Then we make the fitness of all features in $\mathcal{F}_{all} - \mathcal{F}_{\text{eval}=50\%,i}$ to zero. This is done with the assumption that after 50% evaluations useful features should have been used by the collection of trees. Now the fitness and unfitness values of all features in $\mathcal{F}_{\text{eval}=50\%,i}$ are modified according to (2.3.4) and (2.3.5) respectively.

$$F^{50\%,i}_{fitness}(f,i) = \begin{cases} \dfrac{|C^i_f|}{\sum\limits_{\substack{f \neq g \\ g \in \mathcal{F}_{50\%,i}}} |\rho_{fg}|}, & \text{if } f \in \mathcal{F}_{\text{eval}=50\%,i} \\ 0, & \text{otherwise} \end{cases} \tag{2.3.4}$$

$$F^{50\%,i}_{unfitness}(f,i) = e^{-\frac{F^{50\%,i}_{fitness}(f,i) - \min_f\{F^{50\%,i}_{fitness}\}}{\max_f\{F^{50\%,i}_{fitness}\} - \min_f\{F^{50\%,i}_{fitness}\}}}, \tag{2.3.5}$$

where ρ_{fg} is the Pearson's correlation between the f^{th} and the g^{th} features, which is a measure of redundancy. Here we try to select features with higher relevance but reducing the redundancy in the set of selected features. The fitness, defined in (2.3.4), increases when the feature is highly correlated with class label. Similarly, it reduces when it is more correlated with other existent features. This is done to achieve maximum relevance minimum redundancy (MRMR).

After 75% function evaluations, again we take another snapshot of the population. Let the existent features for the population be $\mathcal{F}_{\text{eval=75\%},i} \subseteq \mathcal{F}_{\text{eval=50\%},i}$. Then, the fitness and the unfitness values of the features, in $\mathcal{F}_{\text{eval=75\%},i}$, are defined in (2.3.6) and (2.3.7), respectively.

$$F^{75\%,i}_{fitness}(f,i) = \begin{cases} F^{0\%}_{fitness}(f,i), & \text{if } f \in \mathcal{F}_{\text{eval=75\%},i} \\ 0, & \text{otherwise} \end{cases} \tag{2.3.6}$$

$$F^{75\%,i}_{unfitness}(f,i) = 1.0 - F^{75\%,i}_{fitness}(f,i) \tag{2.3.7}$$

Equation (2.3.6) sets the fitness values of the features that are non-existent in the archive to zero, so that, these features do no get any further chance to be selected during mutation.

2.3.4 Population and Archive Initialization

We initialize each population using ramped-half-and-half method. While constructing the random trees, generation of terminal nodes has been done with a probability p_{var}. To insert a terminal node in a tree, a random number r_n is drawn in $[0, 1]$. If $r_n < p_{var}$ then a feature node is added, otherwise a constant node is added. The function nodes are chosen from the set $\mathcal{G}$, with equal probability of inclusion for all functions. The feature nodes are selected using Roulette wheel selection on fitness $F^{0\%}_{fitness}$, defined in (2.3.2).

To initialize the archive from the initial population, we have used the multiobjective archive initialization scheme present in [116] and [113]. It requires two parameters: maximum archive size (N_{max}) and minimum archive size (N_{min}). To do this, we apply a fast-nondominated-sort [117] on the initial population that contains N_{max} number of solutions and stop the sorting process after obtaining N_{min} number of solutions. To elaborate this, let us assume that the i^{th} front has s_i number of solutions. Then we select all solutions up to the k^{th} front, such that, $\sum_{i=1}^{k-1} s_i < N_{min}$ and $\sum_{i=1}^{k} s_i \geq N_{min}$.

2.3.5 Selection of Crossover or Mutation

Since ASMiGP is a steady state MOGP, in each generation we generate only one offspring. We use either crossover or mutation to do that. The crossover is selected with the probability p_c. A random number r_c is drawn in $[0, 1]$. if $r_c < p_c$ then crossover operator is selected, otherwise the mutation operator is selected for that generation.

2.3.6 Mating Selection

ASMiGP uses crossover with male and female differentiation which needs one male and one female parent. We perform Roulette wheel selection using classification accuracy of the binary classifiers as fitness to select the female parent. Then the male parent is selected randomly from the remaining archive. The only condition to be satisfied is that the male and female parents must be distinct. For mutation, we need only one individual and it is selected in the same way as done for the female parent in case of crossover operator.

A choice of mating selection could have been done using some bi-level selection operator, where Pareto dominance is preferred over diversity. In that case, solutions along the whole Pareto optimal solution set with good diversity would have been selected as the primary (female) parents. Note that, we have used crossover with male-female differentiation that tries to generate an offspring near the primary parent in the hypothesis space. Consequently, this might cause generation of Pareto optimal binary classifiers along the whole Pareto optimal solution set. Though they are Pareto optimal, these binary classifiers may have poor accuracies. This is because of the implicit imbalance nature of the binary classification problems (due to conversion from multi-class classification problems) and different classifier sizes. An ensemble classifier, however, performs better when individual members of the ensemble are more accurate. Therefore, we use classification accuracy based mating selection. It guides the search to generate more accurate binary classifiers.

2.3.7 Crossover

In this study, we have used crossover operation with male and female differentiation. We want the offspring to be near the female (acceptor) parent in the hypothesis space. The male (donor) parent is used to make the offspring diverse from its mother. To do this, two random points (nodes) are selected from each of the parents. The probabilities of selection of terminal nodes and function nodes as a crossover point (node) are respectively p_t^c and $(1 - p_t^c)$. Then, the subtree rooted at the selected node of the mother tree is replaced with a similarly selected subtree from the father tree. If the offspring is identical to any of its parents, the whole procedure is repeated (before evaluation/learning of the offspring).

2.3.8 Mutation

In most GP based systems, for mutation a subtree rooted at a randomly selected node is replaced by a new randomly generated subtree. Though this kind of mutation explores more in the hypothesis space, it may be too disruptive in nature. Therefore, we intend to use less exploration by keeping the tree structure unaltered. During mutation we perform the following operations on a tree: (1) Each constant of the tree is replaced by another random constant with probability p_c^m. (2) Each function node of the tree is replaced by anther random function node with probability p_f^m. (3) Only one feature node of the tree is replaced by another feature node.

For feature nodes, to select the mutation point Roulette wheel selection is performed on the unfitness of the features which are present in the tree. Similarly, to insert a new feature at the mutation point, we select a feature using Roulette wheel selection based on the fitness values (probability proportional to fitness) of the features.

This restricted mutation scheme ensures that the tree structure of an equation remains the same. It also ensures that the variables in an equation do not change drastically - changing more than one variable may shift the solution (equation) in the hypothesis space by a large amount.

2.3.9 Environmental Selection

ASMiGP uses the archive truncation strategy used in [116] and [113]. This multiobjective archive truncation strategy uses Pareto based fitness function, i.e., fitness is Pareto dominance rank. This scheme maintains an archive (ensemble) with an adaptive and dynamic size. It does not allow the archive to fall below a minimum size ensuring diversity in the genotypic space. Moreover, the environmental selection diminishes the exploration in areas of objective space that are less likely to yield improved solutions [113], ensuring diversity in phenotypic space. Furthermore, we have made the following difference in the environmental selection. ASMiGP ensures that the infix expression of every offspring (after mutation or crossover) is distinct from every member of the archive. To achieve this, before evaluating the offspring, ASMiGP converts it to its infix expression, and then, compares the expression with that of each individual of the archive. Only if the expression is unique, the offspring is evaluated and added to the archive. Otherwise, it is discarded and a new offspring is generated. Another noticeable difference is that here the number of objectives is three. Note that the diversity maintenance in each ensemble both in phenotypic space and in genotypic space finds a diverse set of trees (bi-classifiers). Diverse trees enhance the performance of

the corresponding ensemble. In this context, it is worth mentioning that the archive, along with the archive truncation strategy, helps to realize a good Pareto front by explicitly maintaining diversity among the fit (according to rank) solutions. However, the archive alone is not sufficient to evolve a good Pareto front along all the objectives.

2.3.10 Decision Making

To determine the class label, we find the *net belongingness* of a point to each class. The *net belongingness* lies in $[0, 1]$. A higher value indicates more *net belongingness* to that class. A data point is assigned to that class for which it has the highest *net belongingness*.

After the learning, we obtain a set of c archives, $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_c\}$; $\forall i,\ 1 \leq |\mathcal{A}_i| \leq N_{max}$, where c is the number of classes, and $\mathcal{A}_i$ is the set of all binary classifiers for the i^{th} class. To determine the *net belongingness* of a data point $\mathbf{x}$ to class m, $\mathcal{B}_m^{\text{net}}(\mathbf{x})$, it is passed through all genetic programs of $\mathcal{A}_m$; $m = 1, 2, \cdots, c$. The *net belongingness*, $\mathcal{B}_m^{\text{net}}(\mathbf{x})$, of the point $\mathbf{x}$ for class m is computed using (2.3.8).

$$\mathcal{B}_m^{\text{net}}(\mathbf{x}) = \frac{1}{2}\left(\frac{1}{|\mathcal{A}_m|}\sum_{i=1}^{|\mathcal{A}_m|}\mathcal{B}_m^i(\mathbf{x}) + 1.0\right), \tag{2.3.8}$$

where $\mathcal{B}_m^i$ is defined as

$$\mathcal{B}_m^i(\mathbf{x}) = \begin{cases} +\left(1.0 - \dfrac{FP_m^i}{FP_m^{max}}\right), \text{ if } \mathcal{A}_m^i(\mathbf{x}) > 0 \\ -\left(1.0 - \dfrac{FN_m^i}{FN_m^{max}}\right), \text{ otherwise.} \end{cases} \tag{2.3.9}$$

In (2.3.9), FP_m^i and FN_m^i respectively represent the number of FPs and FNs made by the i^{th} individual of $\mathcal{A}_m$ on the training data. FP_m^{max} and FN_m^{max} are respectively the maximum possible FP and the maximum possible FN for the m^{th} class; and $\mathcal{A}_m^i(\mathbf{x})$ is the output from the i^{th} individual of $\mathcal{A}_m$ for input data point $\mathbf{x}$. Finally, $\mathbf{x}$ is assigned the class k, when $\mathcal{B}_k^{\text{net}} = \max_{m=1}^{c}\{\mathcal{B}_m^{\text{net}}\}$. Note that, FP_m^{max} and FN_m^{max} are determined using the training data.

The concept of *net belongingness* is inspired by the concept of negative voting. Negative voting has been widely used in diverse applications [118–120]. It is more effective when circumstances unfavourable to the preferences invoke stronger electoral responses than the similar favourable responses, as well as the behaviours of the voters are well defined [121]. In our scheme, the learners for the i^{th} class learn to vote *yes* for

the points of the i^{th} class and *no* for the points which do not belong to the i^{th} class. For multi-class problems, it is more likely that a binary classifier learns to say *no* than to say *yes* for a larger number of points. Therefore, we found negative voting to be more suitable in this context. However, we have used a weighted negative voting scheme. The accuracies for the responses *yes* and *no* of the i^{th} binary classifier of the m^{th} class are respectively $\left(1.0 - \frac{FP_m^i}{FP_m^{max}}\right)$ and $\left(1.0 - \frac{FN_m^i}{FN_m^{max}}\right)$. These values have been used as corresponding weights for the responses *yes* and *no* of the i^{th} binary classifier of the m^{th} class. We have used the positive and the negative signs to indicate acceptance and rejection of the data point for the m^{th} class, respectively.

2.4 Experiments and Results

2.4.1 Experimental Settings

We have repeated 10-fold cross validation of the proposed method 10 times. Table 2.1 shows the parameter settings that we have used for this purpose. The training data is Z-score normalized. Furthermore, based on the means and the standard deviations of the features of the training data, the test data was Z-score normalized. Note that, except N_{max} and N_{min}, all parameters are standard parameters used in any GP simulations, while N_{max} and N_{min} are needed for archive maintenance. Although all the parameters can be chosen using a cross validation mechanism, because of a huge computational overhead, we could not do that. Based on a few experiments, we have selected these parameters. These choices are not optimal. However, since the same set of values are used for widely different types of data sets, it demonstrates the effectiveness of the proposed scheme. The proposed method has been implemented in Java with the help of jMetal [122, 123].

We have used eight microarray and eleven text data sets which are summarized in Tables 2.2 and 2.3, respectively.

2.4.2 Importance of Feature Selection and Rule Size Reduction

To demonstrate the importance of the proposed FS and rule size reduction scheme, we have compared our method with GP without FS and no restriction to equation size. To do that, we have made the following changes in the proposed method: (1) No explicit FS as described in Section 2.3.3 is done. (2) There are only two objectives, FP and FN. All other parts of the algorithm and the parameter values remain unchanged. Similar

Table 2.1: Parameter Settings for the Proposed Method

Parameter	Value
Set of functions ($\mathcal{G}$)	$\{+, -, \times, \div\}$
Range of initial values of constants ($\mathcal{C}$)	[0, 2]
Maximum depth of tree during initialization	6
Maximum allowable depth of tree	10
Maximum archive size (N_{max})	50
Minimum archive size (N_{min})	30
Initial Probability of feature nodes (p_{var})	0.8
Probability of crossover (p_c)	0.8
Probability of crossover for terminal nodes (p_t^c)	0.2
Probability of mutation for constants (p_c^m)	0.3
Probability of mutation for function nodes (p_f^m)	0.1
Function evaluations for each binary classifier	400000

Table 2.2: Summary of Microarray Data Sets

	Data Set	Features (F)	Samples (S)	Classes (C)	$\left(\frac{F}{S}\right)$
1	Colon	2000	62	2	32.26
2	TOX-171	5748	171	4	33.61
3	Leukemia 1	7129	34	2	209.68
4	Leukemia 2	7129	38	2	187.61
5	CLL-SUB-111	11340	111	3	102.16
6	GCM	16063	144	14	111.55
7	SMK-CAN-187	19993	187	2	106.91
8	GLA-BRA-180	49151	180	4	273.06

Table 2.3: Summary of Text Data Sets

	Data Set	Features (F)	Samples (S)	Classes (C)	$\left(\frac{F}{S}\right)$
1	oh0.wc	3182	1003	10	3.17
2	oh10.wc	3238	1050	10	3.08
3	tr12.wc	5804	313	8	18.54
4	tr23.wc	5832	204	6	28.59
5	tr11.wc	6429	414	9	15.53
6	tr21.wc	7902	336	6	23.52
7	wap.wc	8460	1560	20	5.42
8	ohscal.wc	11465	11162	10	1.03
9	la2s.wc	12432	3075	6	4.04
10	la1s.wc	13195	3204	6	4.12
11	new3s.wc	26832	9558	44	2.81

to the proposed scheme, we have also executed 10-fold cross validation 10 times for all the data sets. The results (mean values of the corresponding 10 runs) obtained using this scheme for microarray and for text data sets are summarized in Tables 2.4 and 2.5, respectively. From these two tables, we observe the following.

Table 2.4: Experimental Results on Microarray Data Sets (Mean Values of 10 Runs of 10-Fold Cross Validation)

Data Set	%TA		FS		TS		(F/T)		%F		$(\%F/T)$	
	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$
Colon	85.12 (2.07)	71.93 (4.12)	195.0	182.5	6.76	123.60	3.11	8.56	9.75	9.13	0.16	0.43
TOX-171	81.07 (3.42)	58.52 (4.57)	407.6	540.9	8.83	148.44	3.53	11.97	7.09	9.41	0.06	0.21
Leukemia1	93.50 (2.85)	74.08 (6.42)	188.6	276.6	3.01	76.09	2.00	6.52	2.65	3.88	0.03	0.09
Leukemia2	91.58 (3.67)	77.33 (6.84)	147.8	229.0	3.50	74.16	2.02	5.72	2.07	3.21	0.03	0.08
CLL-SUB-111	80.52 (2.60)	54.51 (3.51)	335.3	465.3	7.29	134.28	3.20	10.75	2.96	4.10	0.03	0.09
GCM	69.35 (1.96)	33.97 (2.60)	854.3	1086.0	4.70	105.42	1.91	6.22	5.32	6.76	0.01	0.04
SMK-CAN-187	68.68 (1.18)	62.12 (2.30)	319.9	435.8	22.23	135.88	6.41	15.33	1.60	2.18	0.03	0.08
GLA-BRA-180	68.22 (2.25)	58.89 (2.75)	784.4	569.9	11.30	150.81	4.33	11.40	1.60	1.16	0.01	0.02

%TA: Test Accuracy (standard deviation is provided within parenthesis), FS: Number of Features Selected per Classifier, TS: Tree Size, (F/T): Number of Features per Tree, %F: Percentage of Features Selected, (%F/T): Percentage of Features Selected per Tree, PM: Proposed Method, $W_{F\&R}$: GP without FS and Rule Size Reduction

1. In all cases, the test accuracy is much higher for the proposed method. Especially, for GCM having 14 classes, the difference between the test accuracies is remarkably high.

2. The tree size increases significantly when the third objective, i.e., the restriction on rule size, is not used.

3. For most of the data sets, the proposed method selects a smaller number of distinct features per tree as well as per classifier.

These observations clearly demonstrate the importance of the FS as well as constraining the rule size. Our FS scheme discards features with poor relevance and uses features with good discriminating power, yet avoiding use of redundant features. This not only makes the discovery of useful rules easier but also implicitly constrains the rule size. Thus, FS plays a very important role having two positive impacts: it makes identification of useful rules easier, and it promotes the minimization of the third objective.

2.4.3 Comparing with Other Methods

To compare the performance of the proposed method, we have used the experimental results reported in [124] (see Table 4, Table 5, Table 6, and Table 7 of [124]). In [124], the authors have used four different types of classification algorithms: (1) probability based Naive Bayes (NB), (2) tree based C4.5, (3) instance based lazy learning algorithm IB1, and (4) rule based RIPPER, both before and after FS. Along with the full feature

Table 2.5: Experimental Results on Text Data Sets (Mean Values of 10 Runs of 10-Fold Cross Validation)

Data Set	%TA		FS		TS		(F/T)		%F		$(\%F/T)$	
	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$
oh0.wc	87.75 (0.51)	70.53 (1.17)	484.1	2080.7	17.06	145.38	6.46	25.90	15.21	65.39	0.20	0.81
oh10.wc	81.06 (0.61)	66.11 (1.52)	439.6	2166.8	24.11	140.67	7.12	26.03	13.58	66.92	0.22	0.80
tr12.wc	87.86 (1.50)	60.55 (2.36)	603.7	1697.3	8.67	123.91	4.17	17.22	10.40	29.24	0.07	0.30
tr23.wc	93.95 (1.19)	64.90 (2.14)	396.5	1154.2	6.10	117.15	3.09	14.81	6.80	19.79	0.05	0.25
tr11.wc	86.08 (0.72)	64.63 (2.56)	631.6	1785.8	8.82	128.57	4.03	17.03	9.82	27.78	0.06	0.26
tr21.wc	89.64 (1.14)	71.46 (2.62)	362.0	1219.2	9.51	114.51	4.17	15.00	4.58	15.43	0.05	0.19
wap.wc	79.60 (0.60)	58.96 (1.64)	1199.8	4377.5	17.38	125.53	6.07	21.02	14.18	51.74	0.07	0.25
ohscal.wc	73.45 (0.23)	62.59 (1.21)	232.9	3853.8	63.48	111.52	7.78	24.87	2.03	33.61	0.07	0.22
la2s.wc	83.95 (0.62)	67.32 (0.99)	354.6	2369.1	53.17	133.76	11.41	28.65	2.85	19.06	0.09	0.23
la1s.wc	83.06 (0.36)	65.62 (1.05)	380.6	2372.0	56.40	133.21	12.30	28.35	2.88	17.98	0.09	0.21
new3s.wc	81.32 (0.27)	58.96 (1.64)	1674.2	4377.5	33.93	125.53	7.10	21.02	6.24	16.31	0.03	0.08

%TA: Test Accuracy (standard deviation is provided within parenthesis), FS: Number of Features Selected per Classifier, TS: Tree Size, (F/T): Number of Features per Tree, %F: Percentage of Features Selected, (%F/T): Percentage of Features Selected per Tree, PM: Proposed Method, $W_{F\&R}$: GP without FS and Rule Size Reduction

set, they have used six FS algorithms in their experiment: (1) FAST [124], (2) FCBF [125, 126], (3) CFS [127], (4) ReliefF [128], (5) Consist [129], and (6) FOCUS-SF [130]. Use of all features can be viewed as the seventh FS algorithm. To make this paper comprehensive, we are not discussing the experimental settings used in [124]. Note that, accuracies for few data sets for few pairs of FS schemes are not available in [124].

2.4.3.1 Results on Microarray Data Sets

Table 2.4 presents the results of the proposed method on microarray data sets. We have already mentioned that, for each classifier in [124] the authors have used seven FS schemes (six FS methods as well as the set of all features). For each FS method, 10-fold cross validation experiment were repeated five times in [124]. And then, for each FS method, the average accuracy over the five repetitions is reported. We compare this average accuracy with the *average* accuracy that we have obtained by our method over the 10 repetitions of the 10-fold cross validation experiments. In particular, we count the number of cases (each case refers to one FS scheme) in which our algorithm outperforms. Note that, for some combinations of data set and classifier, the total number of cases is less than seven as for those combinations some results are not available. Table 2.6 reports these counts. To elaborate Table 2.6, consider the entry corresponding to column IB1 and row CLL-SUB-111. For the data set CLL-SUB-111, in Table 6 of [124], authors report the performance of the algorithm IB1 for six different FS

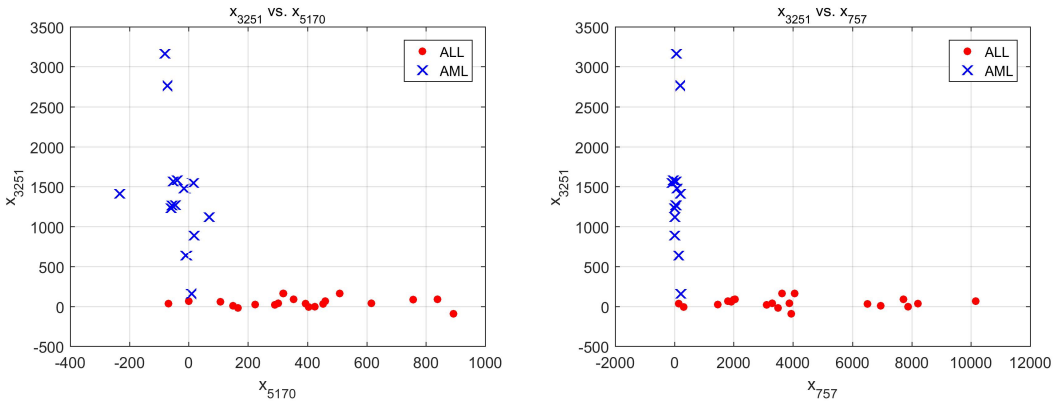
Table 2.6: Comparison with NB, C4.5, IB1, and RIPPER on Microarray Data Sets

Data Set	NB	C4.5	IB1	RIPPER	Total
Colon	4/7	2/7	4/7	6/7	16/28
TOX-171	4/7	7/7	3/7	7/7	21/28
Leukemia1	2/7	2/7	2/7	2/7	8/28
Leukemia2	5/7	6/7	6/7	5/7	22/28
CLL-SUB-111	4/7	6/7	5/6	4/7	19/27
GCM	4/7	7/7	6/7	7/7	24/28
SMK-CAN-187	2/6	2/6	2/6	2/6	8/24
GLA-BRA-180	3/6	5/6	1/6	5/6	14/24
Total	28/54	37/54	29/53	38/54	132/215

algorithms. Our algorithm is found to perform better than five of the six FS algorithms, and hence, the entry for row CLL-SUB-111 and column IB1 is 5/6. For this data set, authors of [124] did not report performance of IB1 using all features. All but the entries in the last column of Table 2.6 are generated in the same manner. The last column of Table 2.6, which reports the row total, reveals that our method performs the best for 61.40% (132 out of 215) test cases.

If we compute the percentage of features selected by the ensembles, it may not be very small for some data sets, like Colon. But, if we consider the number of features selected per binary classifier (tree), we find that this number is quite small, e.g., the maximum value is 0.16% for Colon. We assume that binary classifiers (binary trees) having less than twenty nodes are concise enough. We need to remember that we are talking about raw rules (equations) directly obtained from GP which are most likely affected by bloating. Simplification of these rules may lead to reduction in their sizes. Based on this, in all but one data set we could find easy to analyze rules. For SMK-CAN-187, the extracted rules are more complex possibly because of complex structure of the data.

In Table 2.7, we present the best rules (or binary classifiers or GPs) we found for each class of each microarray data set. These GPs have the highest training accuracy among the GPs obtained from the first fold of the first 10-fold cross validation. If there are more than one rule having the same maximum training accuracy, the rule with the smallest length has been reported. The features in the equations are indexed starting from '0'. From Table 2.7 we can observe that for several classes the proposed method could find substantially small rules. Noticeably, for four, six, eight, and three classes, the proposed method could find the best binary classifier having only one, two, three, and four distinct variables, respectively. So, in 21 out of 33 (i.e., 63.64%) cases, we could find considerably small (and hence easy to interpret) rules. For some classes, the rules having the highest training accuracy were not very comprehensible because



(a) Feature x_{5170} vs. feature x_{3251}

(b) Feature x_{757} vs. feature x_{3251}

Figure 2.1: (a) Feature x_{5170} vs. feature x_{3251} and (b) feature x_{757} vs. feature x_{3251} of Leukemia1 data set.

the length of the rule was not very small. If we think that $(2.0 \times x_i)$ is more easy to understand than $(x_i + x_i)$, though both equations have the same *size*, we observe that for 21 out of 33 (i.e., 63.64%) cases the proposed method could find rules having highest accuracy, which are not affected by bloating and are comprehensible without simplification.

Table 2.7 shows that the rules corresponding to the ALL and AML classes of Leukemia1 data set use two pairs of features: (i) feature x_{5170} and feature x_{3251}, and (ii) feature x_{757} and feature x_{3251}, respectively. To visually inspect the qualities of these two pairs of features, we plot them in Figs. 2.1a and 2.1b, respectively. From Fig. 2.1, we find that these two gene pairs are indeed good. This is because the gene corresponding to x_{3251} is highly expressed for AML but unexpressed for AML, whereas, the other two genes are unexpressed for ALL but expressed for AML.

To investigate how the number of distinct features changes in the archive, we have executed our algorithm with 400000 function evaluations using the entire GLA-BRA-180 data set. In Fig. 2.2, we have plotted the number of distinct features after initial and after each 12.5% of maximum function evaluations. For all four classes of GLA-BRA-180 data set, the number of distinct features in the archive initially falls drastically depicting the impact of FS. After some generations, it becomes almost constant. Moreover, the number of final distinct features for each class is quite small. This indeed reveals a desirable behavior of the our scheme.

To show how the individual binary classifiers' accuracies change with function evolutions, we have plotted each individual's FPs and FNs for all four classes of GLA-

Table 2.7: Some of the Best Genetic Programs for Microarray Data Sets

Data Set	CL	%TrA	Best Genetic Program (Binary Classifier)
Colon	N	98.18	$((x_{1359} + ((x_{1858} + 0.44) - x_{376})) - x_{376})$
	P	98.18	$(((x_{492} - (0.66 + x_{575})) + x_{13}) - x_{1441})$
TOX-171	1	97.39	$((((((-2.16 + (x_{977} + (x_{1327} + x_{1669}))) + x_{1669}) + (x_{5475} + ((x_{2883} + x_{999}) + x_{5568}))) + (x_{2956} + (x_{2956} + (((x_{2429} + -0.49) + x_{1220}) + (x_{1327} + (-2.16 + ((x_{5568} + x_{1220}) + (x_{3601} + x_{999}))))))))) + (x_{2883} + ((-2.16 + x_{2956}) + ((((-2.16 + x_{265}) + x_{1669}) + x_{2883}) + x_{2956}))))$
	2	98.04	$((((((x_{2176} - x_{5608}) + (x_{3527} + -1.48)) - x_{564}) - x_{5608}) + ((x_{5099} + ((x_{521} + ((x_{2176} - x_{5213}) + (x_{685} - 3.04))) + x_{5465})) + x_{4197})) - x_{564})$
	3	100.00	$((x_{624} - (x_{2307} + x_{3601})) - ((x_{2768} + (x_{4073} - (x_{1864} - 1.78))) - ((x_{2015} - 1.08) - x_{4073})))$
	4	99.35	$(((x_{4436} + (((x_{4422} - 2.57) + x_{1983}) + x_{1892})) + x_{5684}) + x_{3440})$
Leukemia1	ALL	100.00	$(x_{5170} - x_{3251})$
	AML	100.00	$(x_{3251} - x_{757})$
Leukemia2	ALL	100.00	$(0.16 - x_{4846})$
	AML	100.00	$(7.6 * x_{4846})$
CLL-SUB-111	1	100.00	$(-1.14 + x_{6559})$
	2	96.97	$((((x_{10124} + x_{10389}) + x_{9520}) - x_{10903}) - ((x_{10092} - (x_{14} - ((x_{8462} - (((x_{10124} - x_{4890}) + x_{4231}) - 1.19)) + x_{6527}))) + x_{6527}))$
	3	96.97	$((x_{4999} - x_{348}) + (((((((x_{64} + x_{1861}) + x_{9138}) + x_{1861}) + ((x_{10178} - x_{8518}) - x_{9448})) + x_{6191}) + x_{5887}) - x_{8518}))$
GCM	0	99.22	$((((((x_{2173} * x_{7307}) - (1.47 + x_{15052})) + ((x_{8462} - 1.81) + (x_{10928} + (x_{11553} - 0.77)))) + x_{8263}) + x_{10928}) + x_{9691})$
	1	100.00	$((x_{12503} * (7.26 - x_{8902})) + x_{6941})$
	2	100.00	$(x_{2435} + ((x_{6867} + (x_{279} - 3.11)) + x_{8333}))$
	3	100.00	$(((x_{3338} - 2.12) + x_{5490}) + x_{12313})$
	4	100.00	$(x_{6656} - 0.15)$
	5	100.00	$((((x_{965} - 1.79) - x_{2715}) - 1.04) - x_{15971})$
	6	98.45	$((x_{2650} - 1.41) - x_{6887})$
	7	100.00	$(x_{3258} + (x_{4136} + (x_{5365} - 2.53)))$
	8	100.00	$((x_{2268} - 1.47) + x_{3596})$
	9	100.00	$(((x_{7067} - 2.23) + x_{3041}) + x_{5274})$
	10	100.00	$((x_{1096} + (x_{15254} - 0.97)) + x_{4454})$
	11	100.00	$((x_{8776} - 3.9) + (x_{2359} + (x_{13518} + x_{14888})))$
	12	100.00	$((-1.48 + x_{13508}) + (x_{2621} \,/\, (-0.87 + x_{7191})))$
	13	100.00	$(x_{11018} + x_{548})$
SMK-CAN-187	1	89.29	$(((((((x_{12065} - x_{3087}) - x_{5094}) + (x_{8465} + x_{10370})) + (((x_{12065} - x_{3087}) + x_{10370}) + x_{16876})) + (x_{16876} - 0.5)) + x_{7209}) + ((((((x_{16876} - (x_{8465} + x_{10370})) - x_{3087}) + (((x_{12065} - x_{3087}) - x_{16876}) + x_{10370})) + (((x_{12065} - 0.29) - x_{5094}) + x_{16876})) + (1.27 + x_{15445})) - (x_{12065} + 0.78)))$
	2	89.29	$(((x_{5701} + ((x_{8889} + ((x_{8889} - x_{11317}) - x_{13749})) + ((0.62 + (1.82 - x_{5569})) - x_{19412}))) - x_{4716}) + ((((x_{15453} \,/\, x_{6993}) * (x_{11521} - x_{7593})) + ((x_{8889} + ((x_{6054} \,/\, x_{6993}) + (((x_{5701} - x_{11041}) - 1.51) - x_{11041}))) - x_{5701})) - x_{4716}))$
GLA-BRA-180	1	100.00	$((x_{36519} - 2.25) + x_{19374})$
	2	96.30	$(((((((((x_{20651} - (x_{43947} - x_{48975})) - 0.94) - (x_{33319} - (x_{20801} - x_{11906}))) + x_{44840}) - (2.64 - x_{15031})) - x_{11906}) + x_{35714}) + (x_{45181} - ((1.78 - (x_{20651} - 0.44)) - (x_{47620} + x_{1685}))))$
	3	94.44	$((((((x_{31170} + x_{27810}) + (((x_{187} + x_{28332}) + (x_{31280} + x_{3320})) + 0.72)) + (x_{187} + x_{28332})) + x_{28332}) + (x_{31280} + (x_{20443} + (x_{31280} + x_{3320}))))$
	4	95.06	$((x_{11225} - 0.78) + ((((((x_{43755} - x_{11522}) - x_{5202}) + (((x_{10207} - x_{10153}) + ((x_{43755} - x_{34025}) + x_{8954})) + x_{29225})) + ((((x_{11225} - 3.46) - x_{30877}) + x_{22932}) + x_{34601})) - x_{5202}) + (((x_{43755} - x_{24511}) + x_{8954}) - x_{5202})))$

CL: Class Label, %TrA: Training Accuracy, N: Negative, P: Positive.

BRA-180 data set in Figs. 2.3 and 2.4. For this part of the experiment also, we have used the entire data set for training. From the figures it is observed that with the

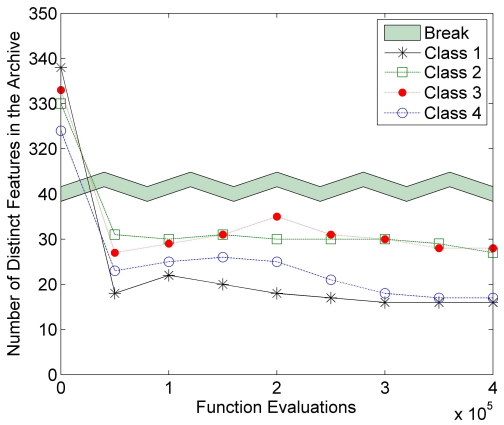


Figure 2.2: The number of distinct features present in the archive with respect to the number of function evaluations for GLA-BRA-180 data set.

increase in the number of evolutions, the average accuracy of the solutions tends to increase. However, for class four, some solutions having lower level of accuracies (having higher FPs and FNs) are there even after 400000 function evaluations. One possible reason for this may be that even after such a high number of function evaluations, the algorithm was still searching for more concise solutions, and could find some trees of comparatively smaller size.

2.4.3.2 Results on Text Data Sets

Similar to the microarray data sets, for the text data sets we present the number of test cases for which our algorithm provides the best results for each classifier and data set pair in Table 2.8. Table 2.8 reveals that for text data sets the proposed scheme performs the best for 95.75% (248 out of 259) cases.

If we consider the same criteria that binary classifiers (trees) having less than twenty nodes are concise enough, then our method could not find concise rules for five (oh10.wc, ohoscal.wc, la2s.wc, la1s.wc, and new3s.wc) out of the eleven text data sets. Some reasons behind this may be that (1) the number of classes is high for the text data sets, and (2) the existence of more complex class structure, which is defined by the keywords and the relation of keywords to the imposed classes is usually not as straightforward as genes have to cancers. However, the percentage of features selected per binary classifiers (trees) is quite small. Noticeably, the accuracy obtained for new3s.wc (having 44 classes), is much higher than that of the other methods (for accuracies of other methods, see [124]). We can also observe that for data sets having more than or equal to ten classes, our method performs comparatively better than

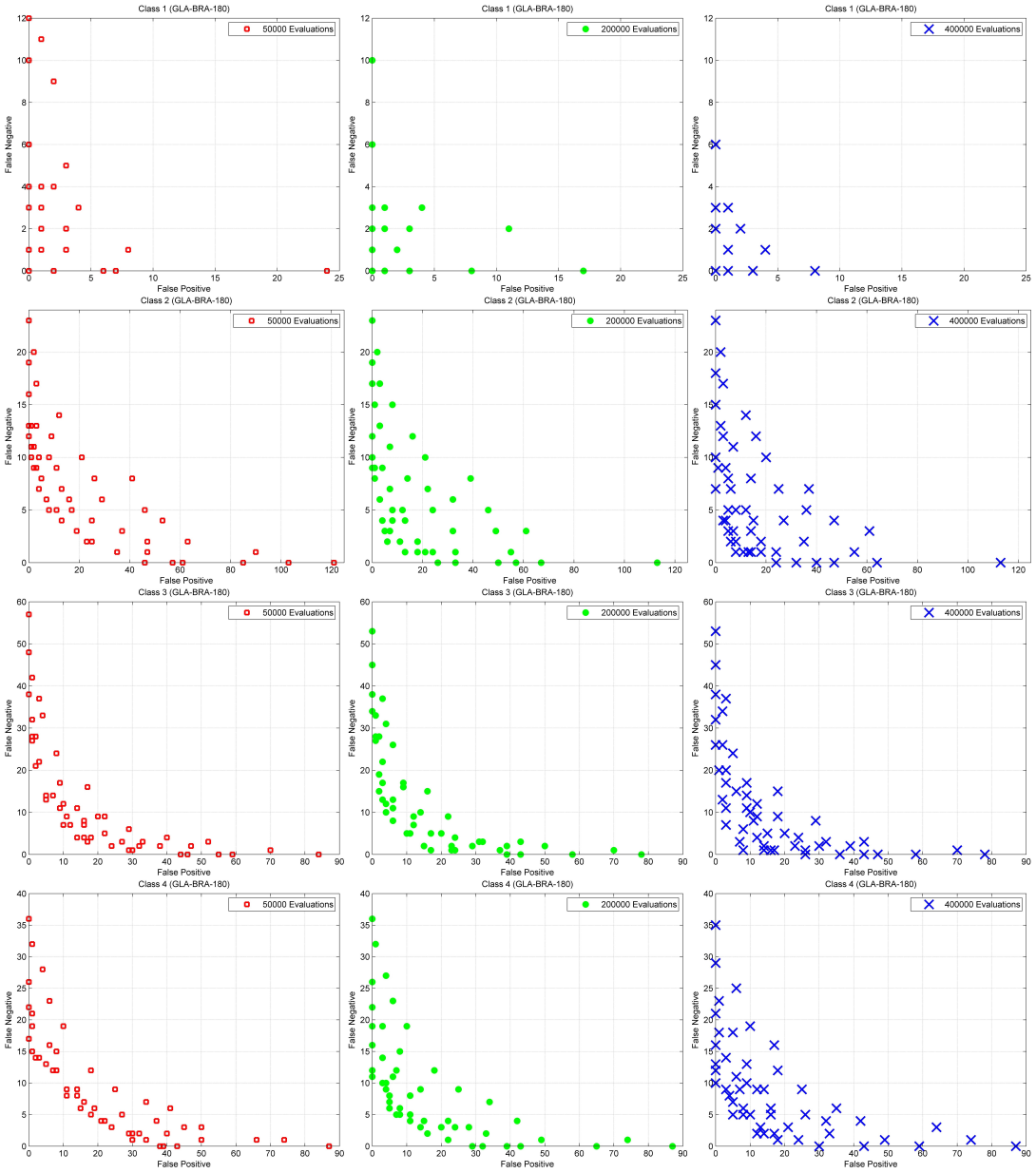


Figure 2.3: Changes in archives with different number of function evaluations for all four classes of GLA-BRA-180 data set (archives at different evaluations are shown separately).

other methods.

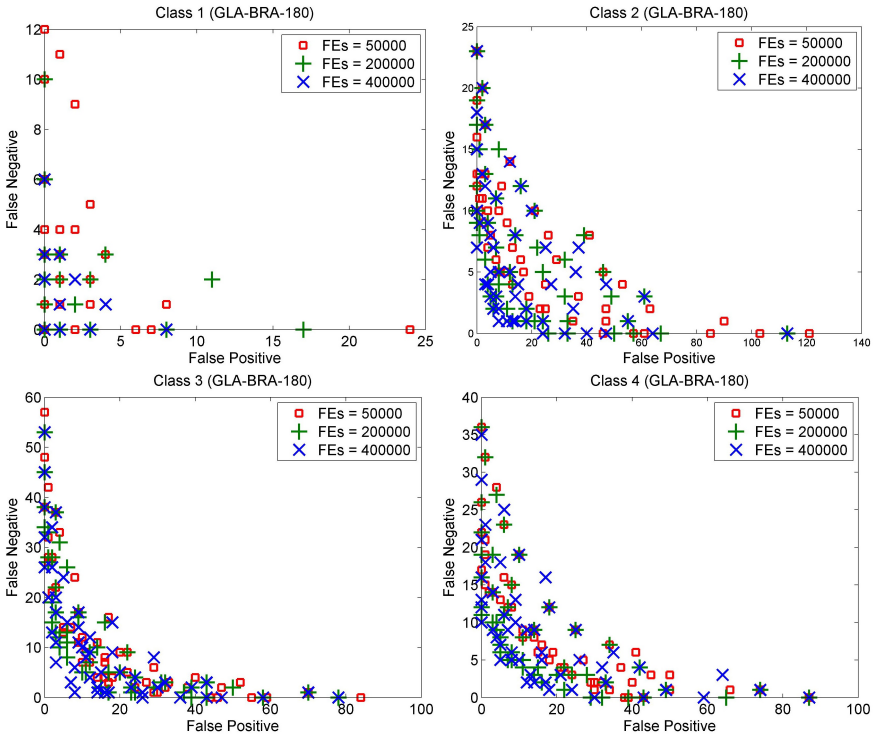


Figure 2.4: Changes in archives with different number of function evaluations for all four classes of GLA-BRA-180 data set.

Table 2.8: Comparison with NB, C4.5, IB1, and RIPPER on Text Data Sets

Data Set	NB	C4.5	IB1	RIPPER	Total
oh0.wc	7/7	7/7	7/7	6/6	27/27
oh10.wc	7/7	7/7	7/7	6/6	27/27
tr12.wc	7/7	7/7	7/7	7/7	28/28
tr23.wc	7/7	5/7	7/7	4/7	23/28
tr11.wc	7/7	7/7	7/7	6/6	27/27
tr21.wc	6/7	6/7	6/7	3/6	21/27
wap.wc	7/7	6/6	6/6	6/6	25/25
ohscal.wc	4/4	4/4	4/4	4/4	16/16
la2s.wc	6/6	5/5	5/5	5/5	21/21
la1s.wc	6/6	5/5	5/5	5/5	21/21
new3s.wc	3/3	3/3	3/3	3/3	12/12
Total	67/68	62/65	64/65	55/61	248/259

2.4.4 Statistical Significance Testing

To compare the proposed method with existing FS and classification methods, we consider four classification and six FS methods as well as with the full feature set. Thus we compare our algorithm with 28 FS and classification algorithm pairs. We have

Table 2.9: Wilcoxon Signed Ranks Test (One-tailed) for Pairwise Comparisons with the Proposed Method at $\alpha = 0.05$

	NB	C4.5	IB1	RIPPER
FAST	A	R	R	R
FCBF	R	R	R	R
CFS	R	R	R	R
ReliefF	R	R	R	R
Consist	R	R	R	R
FOCUS-SF	R	R	R	R
Full Set	R	R	R	R

A: H_0 Accepted, R: H_0 Rejected.

performed Wilcoxon signed ranks test to show that the performance of the proposed algorithm is significantly different over 28 pairs of FS and classification algorithms. For this, we have considered both the microarray and the text data sets together, and have used the average test accuracies achieved with different algorithms on these data sets. Moreover, we have removed the cases for which accuracies are not known (see Table 4, Table 5, Table 6, and Table 7 of [124]). Table 2.9 shows that out of the 28 cases, only in one case the null hypothesis (H_0) is accepted. Here the null hypothesis is that there is no significant difference in performance between our algorithm and a comparing algorithm. We have also used Friedman test [131, 132] to check if all the 29 (28 existing and our proposed) algorithms perform similarly over seven data sets: Colon, TOX-171, Leukemia1, Leukemia2, GCM, tr12.wc, and tr23.wc (see Table 4, Table 5, Table 6, and Table 7 of [124], some accuracies for other data sets are not available). The obtained Friedman statistic is 60.13, which is significant at $\alpha = 0.001$ as the corresponding statistic χ^2 with 28 degrees of freedom is 56.90. Thus Friedman test suggests existence of significant difference among the algorithms.

2.5 Conclusions and Discussions

In this chapter, we have used a multiobjective genetic programming, called ASMiGP, to evolve diverse sets of binary classifiers to solve multi-class classification problems. Simultaneous FS and rule extraction are performed during the genetic evolution. An important objective of this work is to find simple classification rules that may be human understandable, which in the given context translates to rules with simple operations and short length. The proposed method is found to achieve its goal.

Ensembles perform better when weighted voting is used, the members of the ensembles are diverse enough, and the classifiers are accurate [55]. Here, we have cre-

ated c sets of diverse ensembles. Unlike other ensemble based methods, here each ensemble represents diverse classifiers for a single class. The number of features used by the ensembles is high with respect to the number of features selected per (binary classifier) tree. This suggests that the binary classifiers are diverse enough. In our algorithm, we evolve c distinct species in parallel, which try to learn distinct patterns and no inter-species gene exchange is ever allowed. This property makes each species different from the other species.

Our method has been tested on nineteen (eight microarray and eleven text) data sets. The experimental results show that we could find easy-to-understand rules for overall 63.2% (87.5% for microarray and 54.5% for text) data sets. We have compared our method with four classification methods in conjunction with seven FS methods including use of the all-feature set. For 80.17% (61.40% for microarray and 95.75% for text) cases our method outperforms others. Except for one case, the improvement in performance is shown statistically significant compared to the others 27 classification systems.

The overall performance of the proposed method is better on text data sets compared to that of microarray data sets. Two important differences between these two groups of data sets are that (i) microarray data sets have comparatively large feature-to-sample ratios and may not have enough number of points in each class for MOGP to learn the structure of the data sets, and (ii) the text data sets have comparatively large number of features, classes as well as instances. Our limited experiments suggest that when there are enough points in each class so that each species can successfully learn its target pattern, the proposed method works better. We have found that for text data our method performs noticeably better than the other methods, particularly when number of classes is high (ten or more than that).

Chapter 3

Feature Extraction and Selection for Parsimonious Classifiers with Multiobjective Genetic Programming [2]

3.1 Introduction

A support vector machine (SVM) is a binary classifier that finds a separating hyperplane maximizing the margin of separation between the two classes. For a binary classification task, an SVM solves the following optimization problem.

$$\begin{aligned} &\text{minimize} \left\{ ||\mathbf{w}||^2 + \gamma \sum_{k=1}^{n} \xi_k \right\} \\ &\text{subject to} \\ &y_k \left(\mathbf{w} \cdot \mathbf{x}_k + w_0 \right) - 1 + \xi_k \geq 0,\ k = 1, 2, \cdots, n; \\ &\xi_k \geq 0,\ k = 1, 2, \cdots, n; \end{aligned} \tag{3.1.1}$$

where $\mathbf{w} = (w_1, w_2, \cdots, w_d)^T \in \mathbb{R}^d$, $w_0 \in \mathbb{R}$, $y_k \in Y = \{-1, +1\}$, $k \in \{1, 2, \cdots, n\}$, $||\cdot||$ indicates the ℓ^2 norm, and $(\cdot)$ indicates the inner product of two vectors. Here, w_i, $i \in \{1, 2, \cdots, d\}$ is the coefficient of the i^{th} feature, γ is a constant that determines the penalty for a misclassification, and ξ_k, $k \in \{1, 2, \cdots, n\}$ is a slack variable. The human understandability of a two class SVM with a linear kernel is high, because,

it finds out a hyperplane that works as a decision boundary. But, most of the real world data sets are not linearly separable. Therefore, an SVM uses a nonlinear kernel function to project the data into a higher (may be infinite) dimensional space and then finds a hyperplane in that projected space. However, when a non-linear kernel is used, an SVM performs feature extraction (FE) and may (usually will) attain much higher classification accuracy. For example, if we use an RBF kernel or a sigmoidal kernel, the input data are implicitly projected into an infinite dimension. Thus, the use of a non-linear kernel reduces the human understandability of the system. An additional problem is that we cannot realize the importance of the features used in the classifier. Note that, similar statements can be made about the features extracted by many other popular FE techniques, e.g., (deep) neural networks, independent component analysis (ICA), and linear discriminant analysis (LDA). However, if we can explicitly extract some useful features, and then feed them into a linear classifier, (e.g., a linear SVM) simplicity and explainability of the system would be enhanced. In addition to this, depending on the magnitudes of the corresponding weights, we can also assess the importance of the features used in the model. In this work, our aim is to use this concept to design a linear classifier with simple features. Specifically, the objectives of this work are as follows.

1. To investigate the capability of genetic programming (GP) to extract linearly separable features.

2. To propose an integrated methodology using GP and SVM to select and extract linearly separable features for realizing an ensemble of linear binary classifiers. Therefore, the system would be simple and understandable.

To achieve these objectives, we have contributed in the following ways.

1. We have proposed (i) a new encoding of a solution for a GP, (ii) new fitness and unfitness measures for the existing features, (iii) new fitness and unfitness measures for the nodes of a tree using the weights of the corresponding SVM, (iv) new fitness and unfitness measures for the feature-nodes of a tree depending upon the number of occurrences of the features in the archive, (v) a new archive initialization technique that is consistent with the archive maintenance strategy, (vi) an occurrence-based mating selection strategy for mutation, (vii) a weighted crossover technique, and (viii) an occurrence-based mutation technique. Most importantly, we have augmented some new algorithmic components with some existing components to design an integrated methodology using GP and SVM

to extract and select linearly separable features for finding an ensemble of parsimonious classifiers.

2. We have compared the performance of the proposed method with 34 classification systems and have found that the proposed method performs better in 432 out of 570, i.e., 75.79% comparing cases.

3.2 Related Works

There is a large number of references [34,39,40,42,43,45–51,84–94,133–135], where GP has been used for classification, feature selection (FS), and FE. Below we discuss some of them. Detailed literature surveys on this topic can be found in [38,75,136].

Researchers have used GP for designing both binary classifiers [40, 45–47] and multi-class classifiers [34,42,43,48–51,90–92,133,134]. There are several works, where decision trees have been generated using GP [84–89]. Some of these works [87, 88] have used multiobjective genetic programming (MOGP). In another set of works, GP has been extensively used for generating rule-based systems for classification [34, 40, 42, 43, 45–51, 90–92]. GP, moreover, has been used to generate rule-based binary classifiers that can handle imbalanced data [93, 94]. Ensemble based classifiers have also been designed using GP [1, 39], where the individuals of the ensembles participate in voting to predict the class label of a given pattern. A GP based classifier has also been designed for binary image classification, where the number of training instances are limited [135].

GP performs FS and FE implicitly when a classifier is modelled using GP. Different works, however, have explicitly taken care of FS and FE. Here we discuss some of them briefly. Both filter [76–79] and wrapper [80–83] based strategies have been adopted while performing FS using GP. In some works, discriminant function based GP has been used for online FS and multi-class classification [1, 22], where FS is performed in an embedded way. GP has also been used for FE in several works [53,79,137–141]. Additionally, GP has been adopted for FE and FS for classification of high-dimensional data [142]. In [95, 143], GP has been used to extract features for edge detection. Further, both single objective GP and MOGP have been used for image FS to perform segmentation [144].

Here, we propose a discriminant function (DF)-based GP system, and hence, in the reminder of this subsection, we discuss some of the DF-based GP systems that have been used for classification, FS, and/or FE. In [96], researchers have used four different multiobjective approaches into GP to find a set of classifiers maximizing the re-

ceiver operating characteristic convex hull (ROCCH). For furtherance of their method, they [96] have also incorporated two local search techniques, which have been carefully modelled for classification problems. Authors in [96] have also empirically established the effectiveness of the local search strategies in their adopted MOGP schemes. There is another prominent work with convex hull based GP [97], where the differences between the ROCCH maximization problem and the conventional multiobjective problems, have been analyzed. In that work [97], the authors proposed a new GP-based system namely CH-MOGP, that introduces a new selection strategy and a convex hull based sorting. Both of these schemes are developed to address the issues associated with ROCCH maximization problem.

In [145], researchers have proposed a MOGP-based method for FE and data visualization. In [145], each solution is encoded using an array of trees, such that, each tree extracts an individual feature. For the purpose of evolution, they [145] have used six objective functions, three of which correspond to FE, and the remaining three correspond to data visualization. Later, in [53], GP has been used to extract features for regression. Unlike conventional GP that uses the final output of a tree, this method [53] uses features extracted by some subexpressions of the tree. In this regard, they investigated with five different variants considering: (i) only the feature extracted by the root node, (ii) the features extracted by the root node and the leaf nodes, (iii) every subexpression of the tree, (iv) the feature extracted by the root node and all original features, and (v) all the subexpressions of the tree along with all original features. These features are then used in least angle regression [146]. Note that, like [53], in this work also, we use the features extracted by some subexpressions of the trees.

Recently, a new FS algorithm, called GP with permutation importance (GPPI), has been proposed in [52]. The entire process, called GP-GPPI, has two phases. In the first phase, GPPI is used for FS. In this phase, GP individuals with higher *permutation measures* [52], a new measure proposed by them, are selected. In the second phase, another GP-based method is used for symbolic regression. In this phase, only the features selected by GPPI are used.

3.3 Proposed Work

The basic architecture of this work is similar to the work presented in Chapter 2. Here, as done in Chapter 2, we decompose a c-class classification problem into c binary classification problems and then evolve c distinct sets of genetic programs (binary classifiers). Finally, to an unknown pattern $\mathbf{x}$, we assign the class label y_k, if class

Algorithm 3.1 Archive-based steady-state micro genetic programming - 2 (ASMiGP-2)

Initialize the population. Use the initial population to initialize the archive. **while** $Eval^{Count} \leq Eval^{Max}$ **do**
  repeat
    operator = Select crossover or mutation. **if** *operator == crossover* **then**
      Perform crossover with its mating selection.
    end
    else
      Perform mutation with its mating selection.
    end
  until *the infix expression of off-spring is different from the same of any individual of the archive*
  Evaluate the new off-spring. $Eval^{Count} = Eval^{Count} + 1$ Use archive maintenance strategy to update the archive with the new offspring (environmental selection).
end
return the nondominated solutions of the archive.

k attains the maximum *net-belongingness*. The GP based strategy that we use here is called archive-based steady-state micro genetic programming-2 (ASMiGP-2). The basic steps of ASMiGP-2 is provided in Algorithm 3.1. We note that, the objective of the work in Chapter 2 was simultaneous FS and classification using MOGP, while here we have a different set of objectives: (i) to inspect the capability of GP to extract *linearly separable features* and (ii) to develop an integrated methodology using GP and SVM to find an ensemble of classifiers, which are simple, understandable, and parsimonious.

3.3.1 Encoding, Evaluation, and Feature Extraction

Every solution corresponding to the i^{th} binary classification task consists of a binary tree (genetic program) and a binary SVM with a linear kernel. To process a given pattern, at first it is passed through the tree. The outputs from some specific nodes of the tree together comprise the (selected and) extracted features. These features are used in the corresponding SVM. Next we discuss this process with further details.

During the function evaluation, we pass every training pattern through the tree. For every pattern, each node of the tree generates a value. Thus, if there are m nodes in a tree, we obtain m extracted features for every sample in the training data set. If a node appears at the j^{th} position in the post-order traversal of the corresponding tree, we denote the associated value as the j^{th} feature. Now, for the i^{th} binary classification task, we define a target vector $\mathbf{y}^i = (y_1^i, y_2^i, \cdots, y_n^i)^T \in \{-1, +1\}^n$, such that, if the k^{th} pattern belongs to the i^{th} class, $y_k^i = +1$, else $y_k^i = -1$. A binary SVM with a

linear kernel is then trained using the extracted feature set and $\mathbf{y}^i$. The SVM finds out a weight vector $\mathbf{w} = (w_1, w_2, \cdots, w_m)^T \in \mathbb{R}^m$ and a bias $w_0 \in \mathbb{R}$, where w_j is the weight corresponding to the j^{th} feature generated by the j^{th} node. Here, we perform the following trick. Instead of using the entire extracted feature set, if a feature is generated by a constant node or by a function node with a function $\varphi_\pm \in \{+, -\}$, we don't use that feature in training the SVM. We set the weights of such features to zero. Because, a linear SVM finds out a hyperplane that works as the boundary between two classes, a constant feature generated by a constant node cannot help in the process. Moreover, if we consider two feature sets (i) $\mathcal{F}_{a,b} = \{\mathcal{F}_a, \mathcal{F}_b\}$ and (ii) $\mathcal{F}_{a,b,\pm} = \{\mathcal{F}_a, \mathcal{F}_b, \mathcal{F}_\pm\}$, such that $\mathcal{F}_\pm = \mathcal{F}_a \pm \mathcal{F}_b$, then $\mathcal{F}_{a,b,\pm}$ does not help an SVM more than what $\mathcal{F}_{a,b}$ does, because, a hyperplane $w_a x_a + w_b x_b + w_\pm x_\pm + w_0 = 0$ can always be represented by another hyperplane $(w_a + w_\pm)x_a + (w_b \pm w_\pm)x_b + 0 \times x_\pm + w_0 = 0$, where x_ν $(\nu \in \{a, b, \pm\})$ is the variable associated with feature $\mathcal{F}_\nu$. Thus, $\mathcal{F}_{a,b,\pm}$ includes a useless feature $\mathcal{F}_\pm$, which may needlessly increase the computational effort. Consequently, we choose not to use the features extracted by subtrees rooted with a function node with a function $\varphi_\pm \in \{+, -\}$. In this work, we use L_2-regularized L_2-loss support vector classifier in the primal space by minimizing the following Lagrangian.

$$\mathcal{L}_P^{L_2} = ||\mathbf{w}||^2 + \gamma \sum_{k=1}^{n} \left(\max \left\{ 0, 1 - y_k^i \left(\mathbf{w} \cdot \mathbf{x}_k \right) \right\} \right)^2, \tag{3.3.1}$$

where all variables are as defined earlier.

For each solution associated with the i^{th} binary classification task, the FPs and FNs obtained by its SVM on the training data are used along with the number of nodes in its tree as its three objectives. To test a pattern $\mathbf{x}$ by a classifier for the i^{th} class, it $\mathbf{x}$ is passed through its corresponding tree. Then the extracted feature vector is used by its corresponding SVM. Thus, GP is used as a tool for FE.

3.3.2 Fitness and Unfitness Measures for Selection of Features

We use an embedded model of FS, where we perform filtering at the beginning, and after completion of 50% and 75% function evaluations of the corresponding evolutionary learning process. We also perform explicit FS during crossover and mutation. The crossover operation, moreover, performs both implicit and explicit FS on the existing as well as extracted features. To achieve this, we assign fitness values to the features, both fitness and unfitness values to the nodes of the trees, and different unfitness mea-

sures to the leaf nodes of the trees. After that, these measures are used in crossover and mutation to select and extract an enriched feature subset. In the reminder of this subsection, we discuss the filtering mechanism and the fitness and unfitness measures. Later, while discussing the crossover and mutation operations, we discuss how and why these measures are used in the proposed method.

3.3.2.1 Fitness of Features and Filtering Strategy

Here, we use Golub's index [147] to measure the relevance (G_j^i) of the j^{th} feature for the i^{th} classification task:

$$G_j^i = \left(\mu_{ij}^{+1} - \mu_{ij}^{-1}\right) / \left(\sigma_{ij}^{+1} + \sigma_{ij}^{-1}\right), \tag{3.3.2}$$

where $\mu_{ij}^{\mathcal{Y}}$ and $\sigma_{ij}^{\mathcal{Y}}$ denote the mean and standard deviation of the j^{th} feature with class label $\mathcal{Y} \in \{-1, +1\}$ for the i^{th} binary classification task. Note that, a larger value of $|G_j^i|$ denotes a higher discriminating capability of the j^{th} feature for the i^{th} binary classification task.

Let us use $\mathcal{F}_{all}$ to denote the set of all features. For each binary classification task, our intention is to use a subset of $\mathcal{F}_{all}$ that will help the binary classifiers to determine the corresponding binary class label. To attain this, for a given feature set $\mathcal{F}$, using G_j^i, we define the following fitness function for the j^{th} feature as follows.

$$\phi_{\mathcal{F}}^i(j) = \left(|G_j^i| / \left(\max_{k=1}^{|\mathcal{F}|} \{|G_k^i|\}\right)\right)^{\beta}. \tag{3.3.3}$$

Here, β (> 0) is the order of relevance which can be used to tune the FS process. In this work, we consider $\beta = 2$. Note that, G_j^i could be some other measure, such as, correlation or mutual information between the j^{th} feature and the class label. While designing $\phi_{\mathcal{F}}^i(\cdot)$, our intention is to use the relevance of the j^{th} feature for the i^{th} binary classification task.

At the beginning of the evolutionary training process associated with the i^{th} binary classification task, for all features in $\mathcal{F}_{all}$, we find $\phi_{\mathcal{F}_{all}}^i(\cdot)$. All the features with $\phi_{\mathcal{F}_{all}}^i(\cdot) > \tau$ are selected. Here we have used $\tau = 0.35$. Let the selected feature subset corresponding to the i^{th} binary classification task be $\mathcal{F}_{init}^i \subseteq \mathcal{F}_{all}$. To select a new feature in a tree of the initial population or in mutation operations (during the first 50% function evaluations), we select the new feature with a probability proportional to $\phi_{\mathcal{F}_{init}^i}^i(\cdot)$. After 50% function evaluations, we perform an explicit filtering on the ex-

isting features in the i^{th} archive. Let us denote the selected feature subset for the i^{th} binary classification task as $\mathcal{F}^i_{50\%} \subseteq \mathcal{F}^i_{init}$. In the next 25% function evaluations to introduce a new feature in any tree during relevance-based mutation, we select the new feature with a probability proportional to $\phi^i_{\mathcal{F}_{50\%}}(\cdot)$. After 75% function evaluations, we again perform explicit FS (filtering) on the existing features in each archive. Let us denote the selected feature subset for the i^{th} binary classification task as $\mathcal{F}^i_{75\%} \subseteq \mathcal{F}^i_{50\%}$. In the last 25% function evaluations whenever we introduce a new feature node in any tree during relevance-based mutations, we select the new feature with a probability proportional to $\phi^i_{\mathcal{F}_{75\%}}(\cdot)$.

We want to select the most relevant features. So, at the beginning of the search process, we drop the features that are not very useful using a filtering scheme. However, we also want the features to interact with each other so that synergistic set of features that are individually less relevant but as a set has high discriminating power can be found. We assume that after allowing enough interactions, if a feature is not used by any tree in the archive; that feature is not expected to be useful. So, we can discard it. Keeping this in view, we have used two other filtering steps at 50% and 75% function evaluations. The choice of these percentages are intuitive, in fact, the "optimal" choices could be different say 40% and 60%.

Next, we introduce an occurrence-based fitness measure for the features. Let o^i_j be the number of occurrences of the j^{th} feature in the i^{th} archive at a given time. Then, o^i_j is taken as the occurrence-based fitness of the j^{th} feature for the i^{th} binary classification task at that time. To introduce a new feature in a tree during occurrence-based mutation, we select the new feature with a probability proportional to o^i_j.

3.3.2.2 Fitness and Unfitness of Different Nodes

Let there be m nodes in a tree of which m_1 nodes are either constant nodes or are function nodes with a function $+$ or $-$. As already mentioned, features corresponding to these nodes are not used in training the SVMs. The features corresponding to the remaining $m_2 = (m - m_1)$ nodes are used in training the SVM. Hence, associated with the tree, there is a weight vector $\mathbf{w} = (w_0, w_1, w_2, \cdots, w_{m_2})^T \in \mathbb{R}^{m_2+1}$, where w_0 is the bias and w_k, $k \in \{1, 2, \cdots, m_2\}$ is associated with the k^{th} feature of the feature set used to train the SVM. Let, $w_{min} = \min\{w_1, w_2, \cdots, w_{m_2}\}$. The fitness of the j^{th} node of the

tree is defined as

$$\phi_{node}(j) = \begin{cases} |w_j|, \text{if the } j^{th} \text{ node is used in the SVM} \\ \kappa \times w_{min}, \text{otherwise,} \end{cases} \tag{3.3.4}$$

where $\kappa > 0$ is a small constant. Now, we define the unfitness of the j^{th} node as

$$U_{node}(j) = e^{\left(-\frac{\phi_{node}(j) - \min\limits_k \{\phi_{node}(k)\}}{\max\limits_k \{\phi_{node}(k)\} - \min\limits_k \{\phi_{node}(k)\}} \right)}. \tag{3.3.5}$$

We have also used an occurrence-based unfitness measure defined as $1/o_j^i$ for the feature nodes of a tree. We have used this unfitness measure during occurrence-based mutation to select a feature node to be replaced.

3.3.3 Population and Archive Initialization

We have initialized the trees of each solution (corresponding to each population) with N_{init} number of solutions using the ramped-half-and-half algorithm. For constructing the trees randomly, we have selected terminal nodes with a probability p_{var}. To introduce a terminal node in a tree, we have drawn a random number r_n in $[0, 1]$. If $r_n < p_{var}$, then we have added a feature node, otherwise we have added a constant node. We have chosen the function nodes with equal probabilities from the set $\mathcal{G} = \{+, -, \times, \div\}$. We have selected the feature nodes using Roulette wheel selection on fitness $F^i_{\mathcal{F}_{init}}(\cdot)$.

Similar to ASMiGP, ASMiGP-2 uses the same archive maintenance strategy in [113, 116]. This strategy requires three parameters: (i) maximum archive size (N_{max}), (ii) minimum archive size (N_{min}), and (iii) a flag, called *fastNonDominatedSortRequired*. The flag needs to be turned OFF when all the solutions of the archive are nondominated to each other, and ON otherwise. To initialize the archive from the initial population of size N_{init} $(\geq N_{max})$, we perform a fast-nondominated-sorting [117] on the initial population. Let, $\vartheta(i)$ be the number of solutions in the i^{th} sub-front, and $s_k = \sum_{i=1}^{k} \vartheta(i)$. Then, we choose all solutions up to the l^{th} sub-front, such that, $N_{min} \leq s_l \leq N_{max}$ in the archive. However, it may happen that there is no such l. In that case, there will be an l, such that, $s_l < N_{min}$ and $s_{l+1} > N_{max}$. Then, we choose all solutions up to the l^{th} front and the first $(N_{max} - s_l)$ solutions from the $(l+1)^{th}$ front. If $l = 1$, we set *fastNonDominatedSortRequired* OFF, otherwise, we set it ON.

3.3.4 Crossover With Its Mating Selection

We use two types of crossover operators: (i) a random crossover, and (ii) a weighted crossover. We use a mix of these two crossover strategies randomly with equal probabilities. We need a female parent (acceptor) and a male (donor) parent for both crossovers. We perform mating selection for crossover as follows. We select a single solution from the archive with probabilities proportional to their accuracies as the female parent. Then, we choose another random solution as the male parent. Note that, these two parents must be distinct. Next we discuss the crossover strategies.

3.3.4.1 Random Crossover

From each parent, we randomly select a point (node), where p_t^c and $(1 - p_t^c)$ are the probabilities of selection of terminal and nonterminal nodes, respectively. After that, the subtree rooted at the selected node of the father tree is used to replace a similarly selected subtree from the mother tree. We repeat this process before evaluation of the binary solution if the offspring is identical to any of its parents.

3.3.4.2 Weighted Crossover

To select the crossover points, we use Roulette wheel selection on unfitness values $(U_{node}(\cdot))$ and fitness values $(\phi_{node}(\cdot))$ of the nodes of the acceptor parent and donor parent, respectively. After that, with the subtree rooted at the selected node of the donor parent, we replace the subtree rooted at the selected node of the acceptor parent. If the tree of the offspring is identical to any of the solutions present in the population, we repeat the entire process. For a given set of parents, if three successive trials of the weighted crossover fail to generate any such off-spring, we choose the random crossover.

Using the weighted crossover operation, in a single solution we want to gather a useful feature set from the original and extracted features. Therefore, we want to select a node of the acceptor parent, which has contributed less in the binary classification task, to be replaced with a node of the donor parent, which has contributed more in the same task. In the weighted crossover, the selection of a node of the donor parent and another node of the acceptor parent with probabilities respectively proportional to unfitness $(U_{node}(\cdot))$ and fitness $(\phi_{node}(\cdot))$ allows us to do that. Always guiding the search, however, may cause more exploitation and less exploration. Weighted crossover is likely to replace a sub-tree rooted at an unfit node by a sub-tree rooted at a fit node. In that sense it is biased towards exploitation. But random crossover does

not have this bias and provides a high scope of exploration. Use of the both with an equal probability ensures some kind of balance between exploration and exploitation.

3.3.5 Mutation With Its Mating Selection

Here, we use two mutation strategies: (i) a relevance-based mutation, and (ii) an occurrence-based mutation. In the first 50% function evaluations we use only relevance-based mutation, whereas, in the last 50% function evaluations, we use a mix of these two mutations with equal probabilities.

3.3.5.1 Relevance-based Mutation

The objective of this mutation is to make a good tree better by using a randomly selected more relevant (with a higher discriminating power) feature. To achieve this, we select a solution from the archive with a probability proportional to the accuracy of the binary classification task. Then, to perform mutation, we replace each constant node and each function node of the tree by another randomly generated constant node and function nose with probabilities p_c^m and p_f^m, respectively. Next, only a single feature node of the tree is replaced by another feature node. To do this, we select a feature node of the tree with a probability proportional to $U_{node}(\cdot)$. We replace this selected node by a feature node selected with a probability proportional to $\phi^i_{\mathcal{F}}(\cdot)$.

3.3.5.2 Occurrence-based Mutation

The objective of occurrence based mutation is to reject the features that have been used a few times with one that is well used. Let us assume that a solution s has k features and $L(s) = \{l_1, l_2, \cdots, l_k\}$ is the list of indexes of the features present in s. We define the fitness of s as $\min_{l \in L(s)} \left\{ o^i_l \right\}$. We select a solution with a probability proportional to this fitness value. Then, to perform mutation, like the relevance based mutation, we replace each constant node of the tree by another randomly generated constant node with a probability p_c^m. Similarly, each function node is replaced by a function node with a probability p_f^m. After that, we select an unfit node with a probability proportional to $1/o^i_j$. Then, we find a new feature with a probability proportional to o^i_j as a replacement of the unfit feature node.

In the first phase of the evolutionary learning process, we intend to find some good binary classifiers. Therefore, we want each of them to have as many relevant features as possible. Selecting a feature node with a probability proportional to $U_{node}(\cdot)$ is more

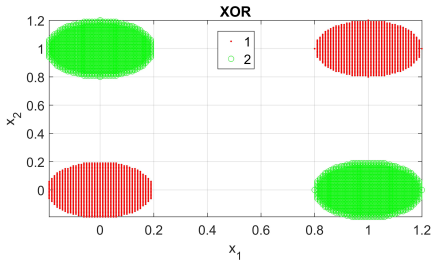
likely to select a feature node that has contributed less in the binary classification task. Again, to accumulate features which are more suitable for the corresponding binary classification task, we select a feature with a probability proportional to $\phi^i_{\mathcal{F}}(\cdot)$. We do this for the first 50% evaluations. For the last 50% evaluations, in addition to selecting features with the maximum relevance, we also intend to reject features that are selected only a few times in the given archive. Consequently, in the last 50% function evaluations, we use a mix of occurrence-based and relevance-based mutations with equal probabilities. We assume that if a feature has a good discriminating capability for the i^{th} class, in the first 50% evaluations, it should be used well by the binary classifiers. Hence, its frequency of occurrences should be high. Note that, though bloating causes some unimportant features to exist in the population, as we use rule size as the third objective, solutions with comparatively smaller trees are more likely to survive. This would cause the number of occurrences of unimportant features to be low.

3.4 Experiments and Results

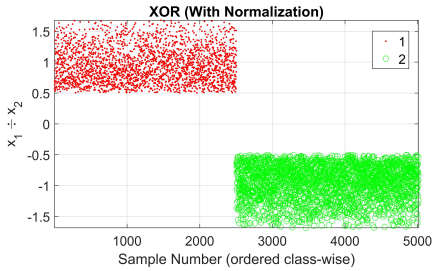
3.4.1 Capability of Genetic Programming to Extract Linearly Separable Features

Here, we investigate the capability of GP to find linearly separable features both with and without normalization of the data. We use three synthetic two class classification data sets. We call these data sets XOR, Disk, and Sphere. Figure 3.1a, 3.2a, and 3.3a show these three data sets, respectively. In the XOR data set, points of the two classes are generated uniformly following an XOR pattern. It is basically an XOR-type data. In Disk data set, the points of the first class are uniformly distributed inside a disk and the points of the second class are uniformly distributed inside a concentric hollow ring. There is a gap between these two classes. In the Sphere data set, the points of the first class are uniformly distributed inside a sphere and the points of the second class are uniformly distributed inside a concentric hollow shell surrounding the sphere. In this data set also, there is a gap between the two classes. Note that, none of these three data sets is linearly separable, and it is not an easy task to extract linearly separable features for them.

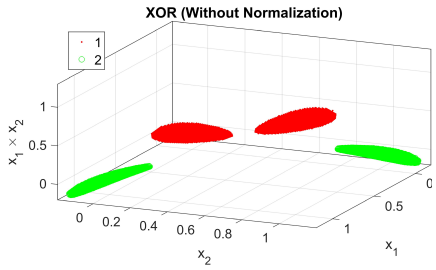
To inspect the effect of data normalization, we have performed Z-score normalization on the original features before beginning of the learning process. During evaluation we also use Z-score normalized values of the extracted features before using them to train the SVM. Furthermore, during prediction of class label of a test data



(a) Original input.

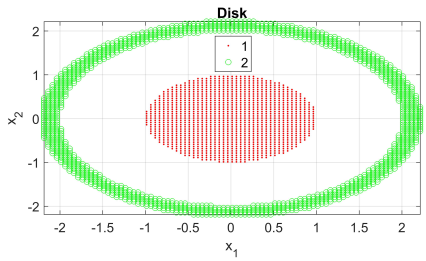


(b) Extracted features (normalized).

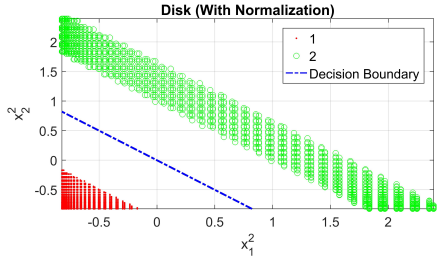


(c) Extracted features (unnormalized).

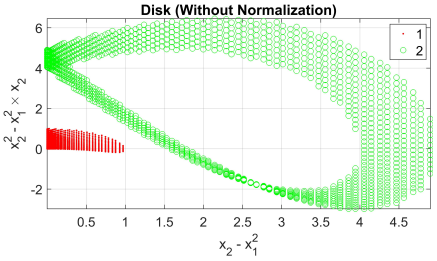
Figure 3.1: The XOR data: the original features and the extracted features with and without normalization.



(a) Original input.

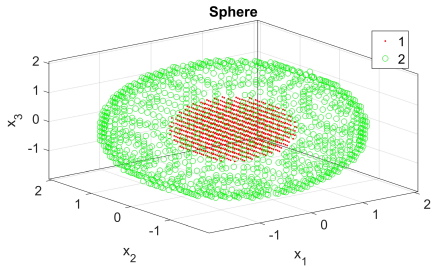


(b) Extracted features (normalized).

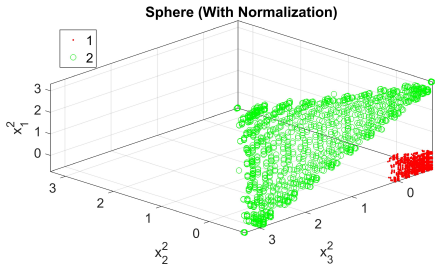


(c) Extracted features (unnormalized).

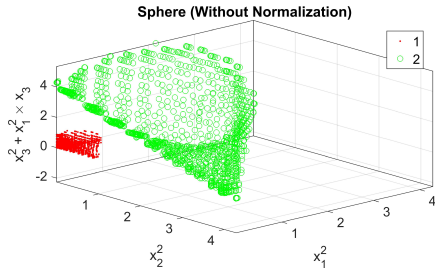
Figure 3.2: The Disk data: the original features and the extracted features with and without normalization.



(a) Original input.



(b) Extracted features (normalized).



(c) Extracted features (unnormalized).

Figure 3.3: The Sphere data: the original features and the extracted features with and without normalization.

point, we also normalize all features using the means and standard deviations of the corresponding feature obtained from the training data. We have also experimented with unnormalized data. For these three data sets, thus, we have six tasks: three with normalization and three without normalization.

We have implemented the proposed method in Java with the help of jMetal [122, 123] and LibLinear [148]. All parameters of the proposed method have been kept the same throughout the experiments. These parameters have been provided in Table 3.1. Note that, most of these parameters are the same as the parameters used in our work discussed in Chapter 2, and hence, Table 3.1 has a resemblance with Table 2.1. As we did in Chapter 2, these parameters have also been chosen based on our initial ad-hoc experiments. The choices of these parameters, consequently, are not optimal. If they are chosen using a cross-validation mechanism, performance of the proposed method may enhance. However, we have not done that due to our limited computing power. The same set of parameter values have been used for a large number of data sets with widely varying type and satisfactory performance has been obtained. Consequently, these values may be considered effective for the proposed method.

Table 3.2 summarizes the results obtained by the proposed method for the synthetic data sets. Table 3.2 displays the accuracies and equations encoded by some

Table 3.1: Parameter Settings for the Proposed Method

Parameter	Value
Set of functions ($\mathcal{G}$)	$\{+, -, \times, \div\}$
Range of initial values of constants ($\mathcal{C}$)	[0, 2]
Maximum depth of tree during initialization	6
Maximum allowable depth of tree	10
Maximum archive size (N_{max})	50
Minimum archive size (N_{min})	30
Initial Probability of feature nodes (p_{var})	0.95
Probability of crossover (p_c)	0.8
Probability of crossover for terminal nodes (p_t^c)	0.2
Probability of mutation for constants (p_c^m)	0.3
Probability of mutation for function nodes (p_f^m)	0.1
Function evaluations for each binary classifier	60000
Order of Relevance (β)	2.0
Threshold of $F_{\mathcal{F}}^i(\cdot)$ (τ)	0.35
Coefficient of minimum weight (κ)	0.1
Cost of misclassification for SVM (`C`) [148]	50
Tolerance of SVM (`eps`) [148]	0.1

specific trees. These trees belong to the most accurate binary solutions (classifiers) of the archive corresponding to the first class. If there are more than one such binary solution, a solution with the minimum tree size is chosen. Along with the accuracies obtained by the proposed method (listed in the second column), Table 3.2 lists the accuracies obtained by these specific binary classifiers (in the last column). It shows that for each task though the proposed method could not find an ensemble with 100% accuracy, it could find at least one linear binary classifier with 100% accuracy. It means that, the proposed method could find at least one linearly separable set of features for each of these six tasks. The surprising simplicity of these extracted features is also worth noting. For each of these six tasks, our intention was to get a set of simple features that can make the classes linearly separable, and the proposed method successfully attains our objective for these six tasks. We have also investigated if there exists any smaller subset of features (among the features selected and extracted by each of these binary classifiers) that can achieve 100% accuracy for the corresponding binary classification task. To visually inspect this, we have plotted all (selected and extracted) features pairs. If there exists no such feature pairs, we have plotted all possible triplets of features. We have found a single linearly separable feature, $(x_1 \div x_2)$, for the normalized XOR data. We have found linearly separable feature pairs for two tasks, Disk with normalization and Disk without normalization. Moroever, we have found linearly separable feature triplets for three tasks, XOR without normalization, Sphere with normalization, and Sphere without normalization. These linearly separa-

Table 3.2: Experimental Results of the Proposed Method for the Artificially Synthesized Data Sets

Task	Accuracy	SMABCC1		
		Equation	ELSFS	Accuracy
XOR (With Normalization)	100.00%	$x_1 \div x_2$	$x_1 \div x_2$	100.00%
Disk (With Normalization)	100.00%	$(x_1 \times x_1) + (x_2 \times x_2)$	$\frac{x_1 \times x_1}{x_2 \times x_2}$	100.00%
Sphere (With Normalization)	96.73%	$(x_2 \times x_2) \div ((x_3 \times x_3) - (x_1 \times x_1))$	$\frac{x_1 \times x_1}{x_2 \times x_2}$ $x_3 \times x_3$	100.00%
XOR (Without Normalization)	100.00%	$x_2 \times x_1$	$\frac{x_1}{x_2}$ $x_2 \times x_1$	100.00%
Disk (Without Normalization)	96.09%	$(x_2 - (x_1 \times x_1)) \times x_2$	$\frac{x_2 - (x_1 \times x_1)}{(x_2 - (x_1 \times x_1)) \times x_2}$	100.00%
Sphere (Without Normalization)	98.83%	$(x_2 \times x_2) \div (x_3 \times (x_3 + (x_1 \times x_1)))$	$\frac{x_1 \times x_1}{x_2 \times x_2}$ $x_3 \times (x_3 + (x_1 \times x_1))$	100.00%

SMABCC1: The Smallest among the Most Accurate Binary Classifiers of Class 1,
ELSFS: Extracted Linearly Separable Feature Subset.

ble subsets of features extracted by the above-mentioned binary classifiers have been listed in the fourth column of Table 3.2.

Let us first consider the normalized version of the XOR type problem. Figure 3.1b shows the scatter plot of the extracted feature $(x_1 \div x_2)$ for this data set. The x-axis represents class-wise ordered sample numbers and the y-axis represents the feature value $(x_1 \div x_2)$. Figure 3.1b immediately reveals that a single feature, $(x_1 \div x_2)$, linearly separates the two classes. We note that, here if $x_2 = 0$, then $(x_1 \div x_2)$ is taken as zero. On the other hand, if we do not normalize the XOR data, the same feature, $(x_1 \div x_2)$, does not make the classes linearly separable (see Fig. 3.4). In that case, our system found the following features: x_1, x_2, and $(x_2 \times x_1)$ (see Table 3.2). Fig. 3.1c depicts that together these three features make the classes linearly separable. Though for the unnormalized XOR data, the chosen tree does not extract any feature or any pair of features that makes the classes linearly separable, it does not mean that there does not exist any such feature. We note here that this set of three features is simpler than a polynomial kernel of degree two. We also note here that for the XOR type data without normalization, GP increased the dimensionality to find linearly separable features.

Next we consider the normalized Disk data, which is not separable using x_1 and x_2. Even for this task, using the same computational protocol, the proposed method found a set of four features: $\{x_1, x_2, x_1^2, x_2^2\}$. Of these four features, x_1^2 and x_2^2 are good enough to make the data set linearly separable as depicted in Fig 3.2b. The separating hyperplane (decision boundary) generated by the SVM for this task is: $-0.85x_1^2 -$

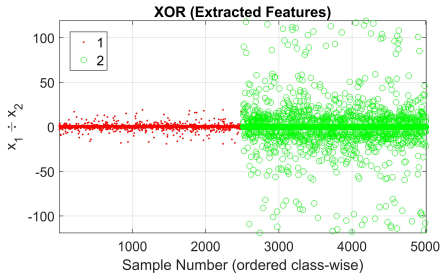


Figure 3.4: Scatter plot of the feature $(x_1 \div x_2)$ for the unnormalized XOR data.

$0.85x_2^2 \approx 0$, which we have depicted in Fig. 3.2b. Here, we mention that the coefficients of feature x_1, feature x_2, and the bias were -0.008, -0.008, and -0.007, respectively, which we have taken as zero. When we do not normalize the Disk data, the proposed algorithm adapts itself using the same protocol to find an appropriate set of features that make the two classes linearly separable. For the tree listed in Table 3.2, extracted feature set $\{(x_2 - x_1^2), (x_2^2 - x_1^2 \times x_2)\}$ makes the two classes linearly separable (Fig. 3.2c).

The Sphere data is a more challenging data. For this, irrespective of whether the data is normalized or not, we found that at least three features extracted by our scheme are needed to make the two classes linearly separable (Table 3.2). For the normalized data the feature set $\{x_1^2, x_2^2, x_3^2\}$ can make the two classes linearly separable (Fig. 3.3b). When the Sphere data is not normalized, the obtained linearly separable feature set is $\{x_1^2, x_2^2, (x_3^2 + x_1^2 \times x_3)\}$, which has been listed in Table 3.2 and illustrated in Fig. 3.3c.

3.4.2 Comparison with Other Methods

3.4.2.1 Data Sets

To compare the proposed method with existing methods, we have used eight microarray and ten text data sets. We have selected these 18 data sets, because, we have used results reported on these data sets in Chapter 2 and [124] for comparison purposes. These 18 data sets have been summarized in Table 3.3. These are all high dimensional data sets and some of them have a few samples, which makes the classifier design challenging. We have performed 10-fold cross validation of the proposed method with the parameters provided in Table 3.1. To reduce the effect of data partitioning and randomness associated with the proposed method, as done in [124] we have repeated 10-fold cross validation five times, and have reported the results in Table 3.4. Here, we have Z-score normalized the data as done for the synthetic data.

Table 3.3: Summary of the Classification Data Sets

Type	Data Set	Features (F)	Samples (S)	Classes (C)	$\left(\frac{F}{S}\right)$
Microarray	Colon	2000	62	2	32.26
	TOX-171	5748	171	4	33.61
	Leukemia 1	7129	34	2	209.68
	Leukemia 2	7129	38	2	187.61
	CLL-SUB-111	11340	111	3	102.16
	GCM	16063	144	14	111.55
	SMK-CAN-187	19993	187	2	106.91
	GLA-BRA-180	49151	180	4	273.06
Text	oh0.wc	3182	1003	10	3.17
	oh10.wc	3238	1050	10	3.08
	tr12.wc	5804	313	8	18.54
	tr23.wc	5832	204	6	28.59
	tr11.wc	6429	414	9	15.53
	tr21.wc	7902	336	6	23.52
	wap.wc	8460	1560	20	5.42
	ohscal.wc	11465	11162	10	1.03
	la2s.wc	12432	3075	6	4.04
	la1s.wc	13195	3204	6	4.12

Table 3.4: Experimental Results of the Proposed Method

Data Set	%TA	FS	TS	$\frac{F}{T}$	$\%F$	$\frac{\%F}{T}$
Colon	81.10 (3.01)	188.4	5.24	3.09	9.42	0.15
TOX-171	79.06 (1.34)	494.2	6.50	3.71	8.60	0.06
Leukemia1	92.33 (1.71)	211.2	3.00	1.98	2.96	0.03
Leukemia2	93.17 (1.37)	354.8	3.00	1.99	4.98	0.03
CLL-SUB-111	77.30 (3.26)	586.8	5.45	3.18	5.17	0.03
GCM	64.03 (2.13)	1895.4	3.77	2.37	11.80	0.01
SMK-CAN-187	71.57 (2.33)	722.8	14.08	7.05	3.62	0.04
GLA-BRA-180	70.33 (1.60)	1000.8	8.75	4.73	2.04	0.01
oh0.wc	88.16 (0.45)	279.0	10.31	5.29	8.77	0.17
oh10.wc	77.98 (0.62)	290.2	15.58	6.10	8.96	0.19
tr12.wc	86.72 (1.70)	452.4	6.57	3.63	7.79	0.06
tr23.wc	93.99 (2.43)	351.8	4.62	2.79	6.03	0.05
tr11.wc	83.68 (0.68)	465.8	7.67	3.90	7.25	0.06
tr21.wc	90.01 (0.85)	529.4	7.46	3.95	6.70	0.05
wap.wc	78.87 (0.90)	1172.2	12.65	6.10	13.86	0.07
ohscal.wc	69.55 (0.09)	182.8	46.79	6.85	1.59	0.06
la2s.wc	78.11 (0.52)	476.6	30.36	11.45	3.83	0.09
la1s.wc	77.03 (0.55)	406.6	33.36	12.03	3.08	0.09

%TA: Test Accuracy (standard deviation is provided within parenthesis), FS: Number of Features Selected per Classifier, TS: Tree Size, $\frac{F}{T}$: Number of Features per Tree, $\%F$: Percentage of Features Selected, $\frac{\%F}{T}$: Percentage of Features Selected per Tree.

3.4.2.2 Comparing Algorithms

We have used the experimental results provided in [124] (see Tables 4, 5, 6, and 7 of [124]) and in Chapter 2 (see Tables 2.4 and 2.5) to compare the performance of the proposed method with the state-of-the-art methods. In [124], the following four classification algorithms have been used: (i) Naive Bayes (NB), (ii) C4.5, (iii) IB1, and (iv) RIPPER. These four algorithms are different in nature: probability based, tree-based, instance based (lazy learning), and rule-based, respectively. In [124], the authors have used these four classification algorithms in conjunction with six FS algorithms along with the full feature sets. The FS algorithms are (i) FAST [124], (ii) FCBF [125, 126], (iii) CFS [127], (iv) RelifF [128], (v) Consist [129], and (vi) FOCUS-SF [130]. We consider the use of full feature set as the seventh FS algorithm. Thus, we have 28 (FS, classification) algorithm pairs. The experimental setup used for these aforementioned algorithms can be found in [124]. As the proposed method is an hybridization of SVM and GP, which uses ensembles of trees (rules), we have also compared the performances of the proposed method with (i) SVM with a linear kernel, (ii) SVM with a RBF kernel, (iii) random forest [149], (iv) AdaBoost [150], (v) tri-objective ASMiGP-based method with embedded FS presented in Chapter 2, and (vi) bi-objective GP-based method without embedded FS discussed in Chapter 2. To compare the performance of the proposed method with SVM with a linear kernel and SVM with an RBF kernel, we have executed the LibSVM [151] implementation of SVM. We have used `gamma` = 1 / number of features, `C` = 1, `cache_size` = 100, `shrinking` = 1 for this experiment (see LibSVM [151] documentation to know more about the parameters). For random forest and AdaBoost, we have used their MATLAB implementations, which are `TreeBagger(·)` and `fitensemble(·)`, respectively. Both parameter `NumTrees` for `TreeBagger(·)` and parameter `NLearn` for and `fitensemble(·)` are set to 200. The results of SVM with linear kernel, SVM with RBF kernel, random forest, and AdaBoost have been provided in Table 3.5. The results of the comparing two GP-based methods have been reported in Tables 2.4 and 2.5.

3.4.2.3 Results and Discussions

In [124], five repetitions of the 10-fold cross validation were performed corresponding to each (FS, classification) algorithm pair. Then, the average accuracy over the five runs were reported for each FS method. We have compared the average accuracies provided in [124] with the average accuracies that we have obtained with the proposed method over the five runs of the 10-fold cross validation (results reported

Table 3.5: Test Accuracies of SVM with Linear Kernel, SVM with RBF Kernel, Random Forest (RF), and AdaBoost

Data Set	SVM[a]		RF[a]	AdaBoost[a]
	Linear Kernel	RBF Kernel		
Colon	81.00 (0.21)	76.86 (1.06)	81.33 (0.00)	77.00 (9.43)
TOX-171	97.11 (1.04)	83.49 (0.79)	78.35 (7.07)	60.12 (0.42)
Leukemia1	92.33 (0.37)	63.67 (0.75)	94.00 (0.00)	60.00 (5.45)
Leukemia2	94.17 (0.37)	71.83 (0.37)	83.50 (3.54)	70.00 (7.07)
CLL-SUB-111	80.58 (1.59)	65.06 (0.07)	74.55 (7.71)	73.64 (1.77)
GCM	68.76 (0.92)	43.39 (0.28)	66.00 (3.03)	51.57 (0.35)
SMK-CAN-187	72.67 (2.39)	68.91 (0.71)	68.00 (2.51)	68.42 (3.31)
GLA-BRA-180	71.44 (0.50)	67.89 (0.25)	71.00 (4.03)	71.57 (2.46)
oh0.wc	80.96 (0.05)	46.64 (0.76)	89.28 (0.28)	63.84 (0.64)
oh10.wc	72.27 (0.98)	46.23 (0.30)	84.19 (1.89)	62.91 (2.35)
tr12.wc	67.10 (1.71)	33.99 (0.14)	87.35 (0.46)	80.00 (1.51)
tr23.wc	71.78 (1.59)	47.21 (0.01)	82.80 (0.71)	92.60 (0.01)
tr11.wc	73.23 (1.61)	37.00 (0.36)	88.20 (2.41)	76.73 (0.61)
tr21.wc	75.11 (0.69)	68.77 (0.03)	86.47 (1.87)	76.47 (0.10)
wap.wc	76.90 (0.06)	46.69 (0.26)	81.15 (1.32)	48.73 (0.17)
ohscal.wc	68.20 (0.08)	60.49 (0.22)	80.86 (0.57)	55.17 (0.11)
la2s.wc	82.50 (0.01)	59.05 (0.16)	88.52 (0.85)	51.04 (1.26)
la1s.wc	81.68 (0.11)	59.44 (0.14)	87.32 (0.45)	52.00 (0.15)

[a]Standard deviations are provided within parenthesis.

in Table 3.4). To be more specific, we have counted the number of cases (each case corresponds to one FS strategy) in which the proposed method has performed better than the comparing method. For some combinations of (FS, classification) algorithm pairs, no result is reported in [124]. Consequently, the total number of comparisons (cases) is less than seven in those cases. We have reported these counts in Table 3.6. To understand this table, let us consider the cell corresponding to the row GLA-BRA-180 and column C4.5. In Table 6 of [124], the accuracies of C4.5 classification scheme were reported for six out of seven FS strategies. The proposed method has outperformed five out of these six FS methods. Consequently, the cell corresponding to column C4.5 and row GLA-BRA-180 contains 5/6. We have reported the column total in the last row of Table 3.6 and the row total in the last column of Table 3.6. The last entry of Table 3.6 indicates that the proposed method outperformed the comparing algorithm in 358 out of 462 cases, i.e., for 77.45% cases. Again, when we compare the proposed method with (i) SVM with linear kernel, (ii) SVM with RBF kernel, (iii) random forest, (iv) AdaBoost, (v) tri-objective ASMiGP-based method with embedded FS presented in Chapter 2, and (vi) bi-objective GP-based method without embedded FS discussed in Chapter 2, it outperforms the comparing algorithms for 10, 17, 6, 17, 6, and 18 cases, respectively. For each of these comparisons there are 18 comparing cases (datasets).

Table 3.6: Comparison with NB, C4.5, IB1, and RIPPER

Data Set	NB	C4.5	IB1	RIPPER	Total
Colon	2/7	0/7	2/7	4/7	8/28
TOX-171	3/7	7/7	3/7	7/7	20/28
Leukemia1	2/7	2/7	2/7	2/7	8/28
Leukemia2	5/7	6/7	6/7	6/7	23/28
CLL-SUB-111	4/7	4/7	4/6	4/7	16/27
GCM	3/7	7/7	4/7	7/7	21/28
SMK-CAN-187	2/6	3/6	5/6	3/6	13/24
GLA-BRA-180	4/6	5/6	2/6	6/6	17/24
oh0.wc	7/7	7/7	7/7	6/6	27/27
oh10.wc	7/7	7/7	7/7	6/6	27/27
tr12.wc	7/7	7/7	7/7	7/7	28/28
tr23.wc	7/7	5/7	7/7	4/7	23/28
tr11.wc	7/7	6/7	7/7	6/6	26/27
tr21.wc	7/7	6/7	6/7	4/6	23/27
wap.wc	7/7	6/6	6/6	6/6	25/25
ohscal.wc	4/4	4/4	4/4	4/4	16/16
la2s.wc	6/6	4/5	4/5	4/5	18/21
la1s.wc	6/6	4/5	5/5	4/5	19/21
Total	90/119	90/116	88/115	90/112	358/462

Thus, in 432 out of 570 comparing cases, i.e, 75.79% cases the proposed method outperforms the comparing algorithms. Except the SVM with a linear kernel, none of the other 33 comparing methods considers a linear classifier or ensembles of linear classifiers, whereas, the proposed methods does that. Consequently, the classifiers modelled with the proposed method are comparatively simpler and more parsimonious.

3.4.2.4 Statistical Significance Testing

At first, we perform Friedman test [131, 132] to check if there exist significant differences among the performances (measured in terms of accuracy) of the comparing 35 (one proposed and 34 existing) methods. Similar to Chapter 2, we use seven data sets for this comparison, because, at least one accuracy corresponding to each of the remaining eleven data sets is not available (see Table 4, 5, 6, and 7 of [124]). These seven data sets are Colon, TOX-171, Leukemia1, Leukemia2, GCM, tr12.wc, and tr23.wc. The test statistic of this experiment is 81.23. This is significant at the significance level $\alpha = 0.001$, because, the associated statistic (χ^2 with 34 degrees of freedom) is 65.24. Consequently, from this experiment we can conclude that there are significant differences among the performances of the comparing algorithms.

Now, we perform pair-wise comparison of the proposed method with these 34

Table 3.7: Wilcoxon Signed Ranks Test (1-tailed) for Pairwise Comparisons with the Proposed Method at $\alpha = 0.01$

CA	C_1F_1	C_1F_2	C_1F_3	C_1F_4	C_1F_5	C_1F_6	C_1F_7	C_2F_1
CO	A	A	A	R	R	R	R	A
CA	C_2F_2	C_2F_3	C_2F_4	C_2F_5	C_2F_6	C_2F_7	C_3F_1	C_3F_2
CO	R	R	R	R	A	R	A	R
CA	C_3F_3	C_3F_4	C_3F_5	C_3F_6	C_3F_7	C_4F_1	C_4F_2	C_4F_3
CO	A	R	R	R	R	A	R	R
CA	C_4F_4	C_4F_5	C_4F_6	C_4F_7	C_5	C_6	C_7	C_8
CO	R	A	A	A	A	R	A	R
CA	C_9	C_{10}						
CO	A	R						

CA: Comparing Algorithm, CO: Comparison Outcome, C_1: NB, C_2: C4.5, C_3: IB1, C_4: RIPPER, C_5: SVM (linear kernel), C_6: SVM (RBF kernel), C_7: Random forest, C_8: AdaBoost, C_9: ASMiGP based method, C_{10}: Bi-objective GP, F_1: FAST, F_2: FCBF, F_3: CFS, F_4: ReliefF, F_5: Consist, F_6: FOCUS-SF, F_7: Full Set, A: H_0 accepted, R: H_0 rejected.

methods. For this purpose, we use Wilcoxon signed ranks test [152] (one-tailed) with the following null hypothesis (H_0): there is no significant difference between the performance of the comparing algorithms. We consider the level of significance (α) as 0.01. We use average accuracy as the performance measure on the 18 data sets. We, moreover, remove the cases for which accuracies were not reported in [124] (see Table 4 5, 6, and 7 of [124]). We have reported the results of these tests in Table 3.7, which shows that, for 20 out of the 34 cases H_0 has been rejected. In other words, in 58.82% cases the performance of the proposed method is significantly better than the comparing algorithms, and for the remaining 41.18% (14 out of 34 cases) cases there is no significant difference between the performance of the proposed method and the comparing methods.

We compare the proposed method and the tri-objective ASMiGP based method in terms of the number of selected features using pairwise Wilcoxon signed ranks test (one-tailed) at the same level of significance (α) and with the same null hypothesis (H_0). In addition to that we also perform the same test with the bi-objective GP based method. In both the cases, H_0 was accepted suggesting that there is no significant difference in terms of the number of selected features at $\alpha = 0.01$. Next we perform Wilcoxon signed ranks test (one-tailed) with the same α and H_0 to compare the rule (tree) sizes of the proposed method with those by both the tri-objective ASMiGP-based method and the bi-objective GP-based method. Here, H_0 was rejected for both the cases suggesting significant differences between the rules sizes of the proposed method and the comparing methods. Since, the proposed method finds rules with

smaller size and the rules extract linearly separable features, the proposed method finds rules with higher parsimony than the comparing GP-based methods.

3.4.3 Effects of New Crossover and Mutation

To investigate the effects of weighted crossover and occurrence based mutation, we consider two variants of the proposed method: $\mathcal{M}_{\mathcal{C}}$ and $\mathcal{M}_{\mathcal{M}}$, which do not use these two operators, respectively. We execute both of these methods on 17 datasets summarized in Table 3.3 except for ohscal.wc (We could not use ohscal.wc due to our limited computing resources). Here, we use the same experimental settings as used for the proposed method. We report the results in Table 3.8. Next, we compare the proposed method (results provided in Table 3.4) with $\mathcal{M}_{\mathcal{C}}$ and $\mathcal{M}_{\mathcal{M}}$ using Wilcoxon signed ranks test (one-tailed). The statistics that we have used for this experiment are the number of features selected per classifier, tree sizes, and test accuracies. When the proposed method is compared with $\mathcal{M}_{\mathcal{C}}$ with these three statistics at statistical significance level $\alpha = 0.05$, we found that the proposed method selected a significantly smaller number of features, generated trees with significantly smaller size, and the test accuracy of the proposed method was significantly better. This experiment validates that the proposed crossover helps in finding smaller/simpler solutions with a smaller number of features. When we compare the proposed method with $\mathcal{M}_{\mathcal{M}}$ at the same level of significance, we observe that the proposed method selected significantly smaller number of features. The other two statistics were comparable at the chosen level of significance. This experiment indicates that the proposed mutation helps in the FS process.

3.5 Conclusions and Discussions

In this Chapter, we have investigated the capability of GP to extract linearly separable features. We have also proposed a multiobjective GP-based embedded FS and FE strategy for generating an ensemble of parsimonious classifiers that intends to use only linearly separable features. For a c class problem, the classifier is an ensemble of c archives, where each archive is an ensemble of binary classifiers. Each binary classifier is composed of a binary tree (genetic program) and an SVM with a linear kernel. In each binary classifier, the genetic program selects and extracts a set of features and the corresponding SVM operates on that feature space.

We have tested the proposed method on three artificial data sets to examine its capability regarding the extraction of linearly separable features. From this experiment,

Table 3.8: Results for Two Variants of the Proposed Method: Without Weighted Crossover ($\mathcal{M}_{\mathcal{C}}$) and Without Occurrence based Mutation ($\mathcal{M}_{\mathcal{M}}$)

Data Set	FS		Tree Size		%TA	
	$\mathcal{M}_{\mathcal{C}}$	$\mathcal{M}_{\mathcal{M}}$	$\mathcal{M}_{\mathcal{C}}$	$\mathcal{M}_{\mathcal{M}}$	$\mathcal{M}_{\mathcal{C}}$	$\mathcal{M}_{\mathcal{M}}$
Colon	196.2	193.2	5.44	5.39	76.71	76.71
TOX-171	488.4	498.0	6.67	6.38	79.75	79.75
Leukemia1	251.6	221.4	3.00	3.00	90.50	90.50
Leukemia2	413.6	339.2	3.00	3.00	93.17	93.17
CLL-SUB-111	628.4	614.4	5.73	5.63	75.80	75.80
GCM	2153.6	1942.8	3.74	3.76	65.47	65.47
SMK-CAN-187	777.8	770.0	14.76	14.36	71.40	71.40
GLA-BRA-180	1070.8	1088.6	9.09	9.00	67.78	67.78
oh0.wc	283.2	276.8	10.63	10.35	88.45	88.45
oh10.wc	293.2	295.4	15.95	15.71	78.42	78.42
tr12.wc	488.4	471.8	6.76	6.63	86.01	86.01
tr23.wc	365.8	369.4	4.61	4.57	94.11	94.11
tr11.wc	465.2	493.4	7.73	7.52	83.72	83.72
tr21.wc	580.6	524.2	7.87	7.57	90.19	90.19
wap.wc	1251.4	1206.2	13.25	12.64	78.31	78.31
la2s.wc	488.4	481.4	33.55	30.05	78.14	78.14
la1s.wc	417.2	415.8	35.77	33.69	77.12	77.12

FS: Number of Features Selected per Classifier, %TA: Test Accuracy.

we have found that the proposed method can successfully extract linearly separable features both with and without data normalization. We have compared the proposed method with 34 state-of-the-art algorithms and found that the proposed method outperforms the comparing methods in 75.79% (432 out of 570) cases. Next, we have performed Friedman test that confirmed the existence of significant differences among the performances of the comparing algorithms. After that, we have performed Wilcoxon signed ranks test for pairwise comparison of the proposed method with the existing methods. This experiment has suggested that the performance of the proposed method is significantly better than 58.82% (20 out of 34) of the comparing methods. These tests have also suggested that there is no significant difference in the performance of the proposed method and the remaining 41.18% (14 out of 34) comparing methods. Consequently, none of the comparing 34 methods has been found to be significantly better than the proposed method. Lastly, we have found that the proposed method can find rules with higher parsimony compared to two GP-based methods, both of which have been discussed in Chapter 2.

Chapter 4

Robust Multiobjective Optimization with Robust Consensus [3, 4]

4.1 Introduction

Group decision making (GDM) problems are frequently encountered in real world problem solving. Depending upon the type of GDM, primarily there are four approaches in the MOP literature [60]:

1. *No-preference*: No apriori preferences for the solutions are provided by the decision makers (DMs). After solving the MOP, the set of obtained solutions is provided to the DMs.

2. *Use of preferences to guide the search (biased)*: The DMs provide their preferences prior to the search, and the preferences are used to guide the search.

3. *Use of preferences after the search*: At first, a multiobjective optimizer is used to find a set of non-dominated solutions. Then, to determine the most suitable solution, an *expert* applies the preferences provided by the DMs.

4. *Use of human interactive refinement during the search*: In this approach, active human intervention is used periodically to refine the obtained solutions and to guide the search. This approach is a hybridization of the previous two approaches.

The posterior approach (the third approach) is less subjective than the apriori and interactive approaches. Therefore, it is probably the most frequently used strategy by the research community [60]. In this work, we try to address two slightly different but related questions. Consider a scenario of robust fuzzy group decision making for a multiobjective optimization problem (FGDM-MOP), where the DMs are weighted, and they provide their preferences either in the objective space or in the variable space. First, if the preferences are available apriori, how can we use these preferences to make the entire search process biased (the second approach) to get solutions that are robust with respect to problem parameters as well as with respect to their degree of consensus? Second, if the preferences are available after the search, how can we assess the qualities of the available solutions (the third approach)? Now, we discuss a real world problem, where we need to answer such questions. Consider the robust multiobjective optimal reactive power dispatch problem [153]. This problem has two objectives: real power loss and voltage deviation. Suppose there are two operators (DMs) with different but fixed preferences about the objectives or the control variables. Here, we are interested in a set of robust solutions with a high degree of consensus. We also note that there is a huge body of literature that uses an iterative consensus reaching process with the help of a moderator. For such methods, the preferences of DMs may change with iterations. This is a different problem. Some of the works, most relevant to our work, can be found in [60, 63, 74]. These works have dealt with either robustness [60, 63] or both robustness and consensus [74] in MOPs. In these works, robust solutions refer to robustness with respect to its variables. In the literature, we could not find any work that searches for robust solutions, which are also robust with respect to consensus among the DMs. The focus of this work is to fill this gap, i.e., find robust solutions which are also robust to their degree of consensus.

Next, we explain the problem using a toy example. Suppose, for a bi-objective optimization problem, there are two DMs with weights $\mathbf{w} = (0.3, 0.7)$. The DMs have provided their preferences in the objective space using Gaussian membership functions as $\mathbf{r}^{\mathcal{O}}_{11} = (100, 10)$, $\mathbf{r}^{\mathcal{O}}_{12} = (110, 5)$, $\mathbf{r}^{\mathcal{O}}_{21} = (95, 12)$, and $\mathbf{r}^{\mathcal{O}}_{22} = (115, 8)$; where $\mathbf{r}^{\mathcal{O}}_{ij} = (c, s)$ denotes a Gaussian membership function with center c and spread s; $i = 1, 2$; $j = 1, 2$. Suppose we have two solutions $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{V}$, such that, $\mathbf{f}(\mathbf{x}_1) = (103, 109)$ and $\mathbf{f}(\mathbf{x}_2) = (93, 118)$. Then, according to the aforementioned preferences, these two solutions have the following membership values: $\mu^{\mathcal{O}}_{11}(\mathbf{x}_1) = 0.96$, $\mu^{\mathcal{O}}_{12}(\mathbf{x}_1) = 0.98$, $\mu^{\mathcal{O}}_{21}(\mathbf{x}_1) = 0.80$, $\mu^{\mathcal{O}}_{22}(\mathbf{x}_1) = 0.75$, $\mu^{\mathcal{O}}_{11}(\mathbf{x}_2) = 0.78$, $\mu^{\mathcal{O}}_{12}(\mathbf{x}_2) = 0.28$, $\mu^{\mathcal{O}}_{21}(\mathbf{x}_2) = 0.99$, and $\mu^{\mathcal{O}}_{22}(\mathbf{x}_2) = 0.93$. The question is how to find a measure using the four membership values associated with each solution to indicate the quality of the solution in terms of consensus. Such a

measure should also be able to compare the solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ quantitatively. Here, we need to aggregate the four membership values associated with each solution in a plausible manner to assess which of the two solutions satisfies both experts to a higher degree or we need to use them to drive the search process.

For a given set of weighted DMs and their preferences, in this work, we have proposed two approaches to define an indicator to measure the consensus of a given solution. We have named this measure *consensus*. For this purpose, we have assumed that preferences are provided either in the objective space or in the variable space using fuzzy numbers. We have further extended these approaches to define an indicator, named *robust consensus*, to measure the robustness of a given solution with respect to its degree of associated consensus. Note that, when there is no perturbation in the system, the proposed definition of *robust consensus* reduces to that of *consensus*. Though there may be countless definitions using the proposed two approaches, we have used them to propose 12 sets of definitions of *consensus*, and the corresponding *robust consensus*. We have also proposed two ways to reformulate a given FGDM-MOP problem for searching a set of solutions that is both robust and enjoys a high degree of consensus. We have discussed the behaviour of the proposed formulations and provided supporting results.

4.2 Related Works

Though the literature on MOPs [1,113,116,154,155], robustness in MOPs [57,60,62–72], GDM [3,156–158], and FGDM-MOP [157] is quite rich, there are only a few works related to FGDM-MOP [3,74], where each DM provides her preference using a fuzzy number [74]. Moreover, there is a huge literature [159–164] in GDM which deals with a different facet of GDM, and hence, their problem formulation is different from ours. There, unlike our formulation, a set of alternative solutions is available and no search process is involved. A set of DMs is involved in the decision making process. Each DM provides her preference using ordering/utility functions/fuzzy preference relation. The consensus reaching process in these works, is iterative, where usually with the help of a moderator the DMs change their preferences to achieve consensus. Note that, in our problem formulation, similar to [74], the preferences provided by the DMs do not change and the preferences help to find the desired solutions via an optimization problem. Due to these differences in the problem formulation and objectives, we choose not to discuss these works further. The only work relevant to ours is due to Xiong et al. [74], which we discuss next.

In [74], Xiong et al. assumed that the i^{th} DM provides her preferences for the j^{th} objective function using a triangular fuzzy number $\mathbf{r}_{ij}^T = \left(r_{ij1}^T, r_{ij2}^T, r_{ij3}^T\right)$. Then, given a set of solutions $\mathcal{S}$, they defined consensus $(\mathrm{cd}(\cdot))$ of the t^{th} solution of $\mathcal{S}$, i.e., $\mathbf{x}_t$, as follows:

$$\begin{aligned}
&\mathrm{cd}(\mathbf{x}_t) = \sum_{i=1}^{d} w_i \rho_i\left(\mathbf{x}_t\right);\\
&\text{where}\\
&\rho_i(\mathbf{x}_t) = \mathcal{A}\left(\varrho_{ij}(\mathbf{x}_t);\ j = 1, 2, \cdots, m\right);\\
&\varrho_{ij}(\mathbf{x}_t) = \frac{\sqrt{\sum_{l=1}^{3} \frac{1}{3} \cdot \left(f_j(\mathbf{x}_t) - r_{ijl}^T\right)^2}}{\left(\max_{k=1}^{|\mathcal{S}|} f_i(\mathbf{x}_k) - \min_{k=1}^{|\mathcal{S}|} f_i(\mathbf{x}_k)\right)}.
\end{aligned} \tag{4.2.1}$$

Here, $\mathcal{A}(\cdot)$ is an aggregation operator, and in [74] authors used it as the arithmetic mean operator. Then, following [60], they [74] proposed a robustness measure $(\mathrm{rd}(\cdot))$ which refers to the principle of *preference robustness*, which we have discussed in Section 1.1.5. Given a non-dominated solution $\mathbf{x} \in \mathcal{V}$ and the set of all non-dominated solutions in its neighbourhood within a radius of δ, denoted by $\mathcal{N}_\delta^{\mathbf{x}}$, $(\mathrm{rd}(\cdot))$ is defined as follows:

$$\mathrm{rd}(\mathbf{x}) = \frac{1}{|\mathcal{N}_\delta^{\mathbf{x}}| + \epsilon} + \frac{\sum_{\mathbf{y} \in \mathcal{N}_\delta^{\mathbf{x}}} \frac{1}{n} \sum_{k=1}^{n} \frac{|x_k - y_k|}{\left(x_k^{max} - x_k^{min}\right)}}{|\mathcal{N}_\delta^{\mathbf{x}}| + \epsilon}, \tag{4.2.2}$$

where ϵ is a small positive value that they [74] considered 1.0×10^{-06}. Here, x_k^{max} and x_k^{min} are respectively the maximum and the minimum values of the k^{th} decision variable in $\mathcal{N}_\delta^{\mathbf{x}}$. Note that, the definition of $\mathrm{rd}(\cdot)$ consists of two components. The first component considers the number of neighbors of $\mathbf{x}$ in $\mathcal{N}_\delta^{\mathbf{x}}$, whereas, the second component computes the average normalized "distance" of $\mathbf{x}$ with its neighbours in $\mathcal{N}_\delta^{\mathbf{x}}$. We note here that in (4.2.1), no membership value is used. It is also not clear, what the denominator of (4.2.1) really represents.

Now, we summarize the innovative points and shortcomings of the prominent works of the literature. In [63], Deb and Gupta defined robustness in MOP both using expectation based and variance based approaches. Later, in [60], Bui et al. defined both dominance robustness and preference robustness in the context of MOPs. How-

ever, none of these two works has dealt with GDM or consensus. The only work that we could find in the literature dealing with both consensus and robustness is in [74]. They introduced a measure of consensus and a measure of robustness, which did not incorporate "robust consensus", i.e., the robustness of a robust solution with respect to its degree of consensus. Here, our objective is to find robust optimal solutions with robust consensus [3].

4.3 Proposed Work

In Section 4.1, using a toy example we found that there is a need for aggregation operators to define consensus. Here, first we talk about some useful aggregation operators, and then, use them to define consensus.

4.3.1 Common Aggregation Operators

Here, we discuss three aggregation operators that we have extensively used in this work. As inputs, each operator takes a set of arguments (membership values or degree of satisfaction of some property) $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_d)$ and a set of weights $\mathbf{w} = (w_1, w_2, \cdots, w_d)$ that are associated with the arguments, such that, $\forall_i\, w_i \in (0, 1)$ and $\sum_{i=1}^{d} w_i = 1$.

The first operator [165] is a weighted conjunction operator, defined as follows.

$$\psi_C(\boldsymbol{\alpha}, \mathbf{w}) = \min_{i=1}^{d} \left\{ \max \left\{ \left(1 - \frac{w_i}{\max_{k=1}^{d} \{w_k\}} \right), \alpha_i \right\} \right\}. \tag{4.3.1}$$

The second operator is a weighted T-norm operator [166] as defined bellow.

$$\psi_T(\boldsymbol{\alpha}, \mathbf{w}) = h^{-1}\left(\sum_{i=1}^{d} w_i \cdot h(\alpha_i) \right), \tag{4.3.2}$$

where $h(\cdot)$ is the generating function of any continuous Archimedean T-norm operator and $h^{-1}(\cdot)$ is the pseudo-inverse of $h(\cdot)$. In this study, we choose $h(z) = -\log(z)$, i.e., $h^{-1}(z) = e^{-z}$, and $\psi_T(\boldsymbol{\alpha}, \mathbf{w}) = \prod_{i=1}^{d} \alpha_i^{w_i}$ [166].

The third operator is a weighted arithmetic mean operator defined below in (4.3.3).

$$\psi_M(\boldsymbol{\alpha}, \mathbf{w}) = \sum_{i=1}^{d} w_i \cdot \alpha_i. \tag{4.3.3}$$

Throughout this work, $\psi(\cdot)$ is used to denote any of the operators $\psi_C(\cdot)$, $\psi_T(\cdot)$, and $\psi_M(\cdot)$.

4.3.2 Consensus

We define consensus using two approaches. Both approaches are applicable irrespective of whether the preferences are provided in the objective space or in the variable space. However, in this subsection, to enhance the understandability, we assume that the preferences are provided in the objective space. Later, we have mentioned how the proposed approaches can be applied, if preferences are provided in the variable space.

For a given solution $\mathbf{x} \in \mathcal{V}$, let $\mathcal{U}_{\mathbf{x}} = \left[u_{ij}\right]_{d\times m} \in [0,1]^{d\times m}$ be a matrix, such that, $u_{ij} = \mu_{ij}^{\mathcal{O}}(\mathbf{x})$, i.e., u_{ij} is the degree to which $f_j(\mathbf{x})$ satisfies the preference provided by the i^{th} DM for the j^{th} objective. Then, we define the degree of satisfaction of the i^{th} DM for $\mathbf{x}$ as follows.

$$\sigma_i(\mathbf{x}) = \phi\left(u_{i1}, u_{i2}, \cdots, u_{im}\right); i = 1, 2, \cdots, d; \tag{4.3.4}$$

where $\phi(\cdot)$ is any aggregation operator, which usually is a T-norm or the mean$(\cdot)$ operator applied over the degree of satisfaction of preferences for all m objectives, $f_j(\mathbf{x}); j = 1, 2, \cdots, m$. For simplicity, we choose min$(\cdot)$ as the T-norm operator throughout this work. Consequently, unless stated explicitly, $\phi(\cdot)$ denotes either the min$(\cdot)$ or the mean$(\cdot)$ operator. Similarly, for $\mathbf{x}$, we define the degree to which all DMs are satisfied, i.e., the level of consensus on the j^{th} objective $f_j(\mathbf{x})$ corresponding to $\mathbf{x}$, as follows.

$$\gamma_j(\mathbf{x}) = \psi\left(\left(u_{1j}, u_{2j}, \cdots, u_{dj}\right), \mathbf{w}\right); j = 1, 2, \cdots, m; \tag{4.3.5}$$

where, as mentioned earlier, $\psi(\cdot)$ is one of the operators $\psi_C(\cdot)$, $\psi_T(\cdot)$, and $\psi_M(\cdot)$. Now we define consensus using two approaches.

1. *Approach I*: We define the level of consensus (overall satisfaction) on $\mathbf{x}$ as follows.

$$\mathcal{C}^1(\mathbf{x}) = \psi\left(\left(\sigma_1, \sigma_2, \cdots, \sigma_d\right), \mathbf{w}\right). \tag{4.3.6}$$

2. *Approach II*: We define the level of consensus (overall satisfaction) on $\mathbf{x}$ as follows.

$$\mathcal{C}^2(\mathbf{x}) = \phi\left(\gamma_1, \gamma_2, \cdots, \gamma_m\right). \tag{4.3.7}$$

Table 4.1: Different Definitions of Consensus using Approach I ($\mathcal{C}^1(\cdot)$) with Different Choices of $\phi(\cdot)$ and $\psi(\cdot)$

		$\phi(\cdot)^\dagger$	
		$\min(\cdot)$	$\text{mean}(\cdot)$
$\psi(\cdot)^\ddagger$	$\psi_C(\cdot)$	$\mathcal{C}^1(\cdot)\vert_{\phi(\cdot)=\min(\cdot),\psi(\cdot)=\psi_C(\cdot)}$	$\mathcal{C}^1(\cdot)\vert_{\phi(\cdot)=\text{mean}(\cdot),\psi(\cdot)=\psi_C(\cdot)}$
	$\psi_T(\cdot)$	$\mathcal{C}^1(\cdot)\vert_{\phi(\cdot)=\min(\cdot),\psi(\cdot)=\psi_T(\cdot)}$	$\mathcal{C}^1(\cdot)\vert_{\phi(\cdot)=\text{mean}(\cdot),\psi(\cdot)=\psi_T(\cdot)}$
	$\psi_M(\cdot)$	$\mathcal{C}^1(\cdot)\vert_{\phi(\cdot)=\min(\cdot),\psi(\cdot)=\psi_M(\cdot)}$	$\mathcal{C}^1(\cdot)\vert_{\phi(\cdot)=\text{mean}(\cdot),\psi(\cdot)=\psi_M(\cdot)}$

†First operator, applied row-wise on $\mathcal{U}_\mathbf{x}$.
‡Second operator, applied column-wise on the obtained set of values computed with the first operator.

Table 4.2: Different Definitions of Consensus using Approach II ($\mathcal{C}^2(\cdot)$) with Different Choices of $\psi(\cdot)$ and $\phi(\cdot)$

		$\psi_(\cdot)^\dagger$		
		$\psi_C(\cdot)$	$\psi_T(\cdot)$	$\psi_M(\cdot)$
$\phi(\cdot)^\ddagger$	$\min(\cdot)$	$\mathcal{C}^2(\cdot)\vert_{\psi(\cdot)=\psi_C(\cdot),\phi(\cdot)=\min(\cdot)}$	$\mathcal{C}^2(\cdot)\vert_{\psi(\cdot)=\psi_T(\cdot),\phi(\cdot)=\min(\cdot)}$	$\mathcal{C}^2(\cdot)\vert_{\psi(\cdot)=\psi_M(\cdot),\phi(\cdot)=\min(\cdot)}$
	$\text{mean}(\cdot)$	$\mathcal{C}^2(\cdot)\vert_{\psi(\cdot)=\psi_C(\cdot),\phi(\cdot)=\text{mean}(\cdot)}$	$\mathcal{C}^2(\cdot)\vert_{\psi(\cdot)=\psi_T(\cdot),\phi(\cdot)=\text{mean}(\cdot)}$	$\mathcal{C}^2(\cdot)\vert_{\psi(\cdot)=\psi_T(\cdot),\phi(\cdot)=\text{mean}(\cdot)}$

†First operator, applied column-wise on $\mathcal{U}_\mathbf{x}$.
‡Second operator, applied row-wise on the obtained set of values computed with the first operator.

Note that, if $\phi(\cdot)$ is chosen as any T-norm operator, for instance $\min(\cdot)$, and $\psi(\cdot)$ is chosen as either $\psi_C(\cdot)$ or $\psi_T(\cdot)$, both definitions of consensus, i.e., $\mathcal{C}^1(\cdot)$ and $\mathcal{C}^2(\cdot)$ become the strictest ones. In that case, if there is no region that is common to every DMs' choice, there would be no solution with nonzero consensus. In other words, $\forall \mathbf{x} \in \mathcal{V}, \mathcal{C}^1(\mathbf{x}) = 0$ and $\mathcal{C}^2(\mathbf{x}) = 0$ would hold true.

Using this framework (Approach I and Approach II) countless definitions of consensus can be generated. However, in this work, we restrict ourselves to $\phi(\cdot)$ as $\min(\cdot)$ or $\text{mean}(\cdot)$; and $\psi(\cdot)$ as $\psi_C(\cdot)$, $\psi_T(\cdot)$ or $\psi_M(\cdot)$. Thus, using Approach I, with different choices of $\phi(\cdot)$ and $\psi(\cdot)$, we generate six definitions of consensus ($\mathcal{C}^1(\cdot)$). They are provided in Table 4.1. Similarly, using Approach II, with different choices of $\psi(\cdot)$ and $\phi(\cdot)$, we generate six definitions of consensus ($\mathcal{C}^2(\cdot)$), which are provided in Table 4.2.

If $w_i = 1/d; i = 1, 2, \cdots, d$; then $\psi_C(\cdot) = \min(\cdot)$ holds. Consequently, Approach I with $\phi(\cdot) = \min(\cdot)$ and $\psi(\cdot) = \psi_C(\cdot)$ is the same as the Approach II with $\psi(\cdot) = \psi_C(\cdot)$ and $\phi(\cdot) = \min(\cdot)$. In this case, $\forall \mathbf{x} \in \mathcal{V}, \mathcal{C}^1(\mathbf{x}) = \mathcal{C}^2(\mathbf{x}) = \min(u_{ij}; i = 1, 2, \cdots, d; j = 1, 2, \cdots, m)$ holds true. Similarly, if $w_i = 1/d; i = 1, 2, \cdots, d$; then $\psi_M(\cdot) = \text{mean}(\cdot)$. Thus, $\forall \mathbf{x} \in \mathcal{V}, \mathcal{C}^1(\mathbf{x})\vert_{\phi(\cdot)=\text{mean}(\cdot),\psi(\cdot)=\psi_M(\cdot)} = \mathcal{C}^2(\mathbf{x})\vert_{\psi(\cdot)=\psi_M(\cdot),\phi(\cdot)=\text{mean}(\cdot)} = \text{mean}(u_{ij}; i = 1, 2, \cdots, d; j = 1, 2, \cdots, m)$ holds true. Note that, $\phi(\cdot)$ can be any T-norm. Consequently, it can be chosen as the product of all its arguments, i.e., $\phi(\alpha_1, \alpha_2, \cdots, \alpha_m) =$

$\prod_{j=1}^{m} \alpha_j$. If we consider $\phi(\cdot)$ as the product T-norm and $\psi(\cdot) = \psi_T(\cdot)$, then $\mathcal{C}^1(\mathbf{x}) = \mathcal{C}^2(\mathbf{x}) = \prod_{i=1}^{d} \prod_{j=1}^{m} u_{ij}^{w_i}$. There may be other cases too when $\mathcal{C}_1(\cdot) = \mathcal{C}^2(\cdot)$ may hold true.

Note that, in Approach II, the weights have a stronger influence on the computed consensus in the sense that in Approach II, every membership value is modulated by the weights, and those modulated membership values are aggregated. On the other hand, in Approach I, after we aggregate the degree of satisfaction of the objectives using an aggregation operator, we use the weights. The nature of influence of weights will depend on the particular choice of $\psi(\cdot)$. For example, if we use $\psi(\boldsymbol{\alpha}, \mathbf{w}) = \psi_T(\boldsymbol{\alpha}, \mathbf{w}) = \prod_{i=1}^{n} \alpha_i^{w_i}$, then every modulated value of α_i is increased, and for a fixed weight, smaller membership values are increased relatively more. On the other hand, if we use $\psi(\cdot) = \psi_M(\cdot)$, every value is reduced, in particular, reduction in α_i is proportional to $(1 - w_i)$.

If the preferences are provided in the variable space, consensus can be defined in a similar manner. The only difference is that the parameter m would be replaced by the parameter n, i.e., the dimension of $\mathcal{U}_{\mathbf{x}}$ would be $(d \times n)$ and there would be n input parameters to the $\phi(\cdot)$ operators used in (4.3.4) and (4.3.7).

Let us revisit the toy example of consensus that we discussed in Section 4.1. We can define a matrix for $\mathbf{x}_1$ as

$$\mathcal{U}_{\mathbf{x}_1} = \begin{bmatrix} \mu_{11}^{\mathcal{O}}(\mathbf{x}_1) & \mu_{12}^{\mathcal{O}}(\mathbf{x}_1) \\ \mu_{21}^{\mathcal{O}}(\mathbf{x}_1) & \mu_{22}^{\mathcal{O}}(\mathbf{x}_1) \end{bmatrix} = \begin{bmatrix} 0.96 & 0.98 \\ 0.80 & 0.75 \end{bmatrix}. \tag{4.3.8}$$

Let us select $\phi(\cdot) = \min(\cdot)$. Then, we can compute the degree of satisfaction for the DMs as $\sigma_1(\mathbf{x}_1) = \min(0.96, 0.98) = 0.96$, and $\sigma_1(\mathbf{x}_1) = \min(0.80, 0.75) = 0.75$. If we choose $\psi(\cdot) = \psi_T(\cdot)$, using Approach I, we can define consensus of $\mathbf{x}_1$, i.e., $\mathcal{C}^I(\mathbf{x}_1) = 0.96^{0.3} \times 0.75^{0.7} = 0.81$. Similarly, we can find $\mathcal{C}^I(\mathbf{x}_2) = 0.68$. As $\mathcal{C}^I(\mathbf{x}_1) > \mathcal{C}^I(\mathbf{x}_2)$, we can conclude that solution $\mathbf{x}_1$ is better than solution $\mathbf{x}_2$ in terms of consensus.

4.3.3 The Proposed Definition of Consensus may Fail in Robust Optimization

A robust solution, which has a good degree of consensus measured either in terms of $\mathcal{C}^1(\cdot)$ or $\mathcal{C}^2(\cdot)$, may not be equally robust with respect to its consensus. We consider Fig. 4.1 to discuss this issue with an example. In the left panel, Fig. 4.1 shows a robust solution $\mathbf{x} \in \mathcal{V}$ in the variable space along with a neighborhood $\mathcal{B}_{\delta}^{\mathcal{V}}(\mathbf{x}) \in \mathcal{V}$. In the right panel, Fig. 4.1 shows the objective vector corresponding to $\mathbf{x}$, i.e., $\mathbf{f}(\mathbf{x}) \in \mathcal{O}$, in the

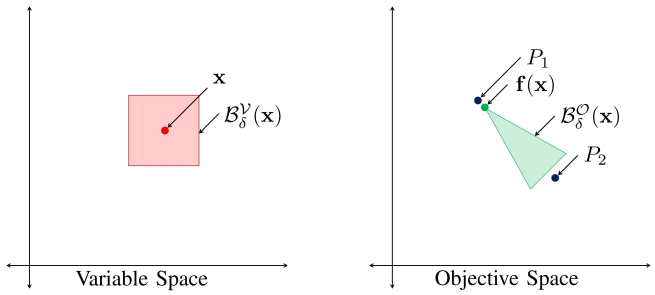


Figure 4.1: In the left panel, showing $\mathbf{x} \in \mathcal{V}$ and $\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x}) \in \mathcal{V}$ in the variable space. In the right panel, showing $\mathbf{f}(\mathbf{x}) \in \mathcal{O}$ $\mathcal{B}_\delta^{\mathcal{O}}(\mathbf{x}) \in \mathcal{O}$, and the preferences P_1 and P_2 in the objective space, such that, $\forall \mathbf{y} \in \mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x}), \mathbf{f}(\mathbf{y}) \in \mathcal{B}_\delta^{\mathcal{O}}(\mathbf{x})$.

objective space along with a neighborhood $\mathcal{B}_\delta^{\mathcal{O}}(\mathbf{x}) \in \mathcal{O}$, such that, $\forall \boldsymbol{y} \in \mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x}), \mathbf{f}(\mathbf{y}) \in \mathcal{B}_\delta^{\mathcal{O}}(\mathbf{x})$. We note here that, $\mathcal{B}_\delta^{\mathcal{O}}(\mathbf{x})$ may not necessarily be a convex set like in the right panel figure. There are two DMs, who have provided their preferences as CLOSE to P_1 and CLOSE to P_2, respectively. We also assume that the weight vector associated with the DMs is $\mathbf{w} = (w_1, w_2)$, such that, $w_1 >> w_2$. When there is no perturbation in the system, $\mathcal{C}^1(\mathbf{x})$ and $\mathcal{C}^2(\mathbf{x})$ would be high, because, $\mathbf{f}(\mathbf{x})$ is closer to P_1 and $w_1 >> w_2$. However, if $\mathbf{x}$ is perturbed in $\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})$, $\mathcal{C}^1(\mathbf{x})$ and $\mathcal{C}^2(\mathbf{x})$ may not remain high. Because, due to the perturbations in the variable space, $\mathbf{f}(\mathbf{x})$ may get shifted away from P_1, and hence, consensus measured in terms of $\mathcal{C}^1(\cdot)$ and $\mathcal{C}^2(\cdot)$ may decrease. In this case, though $\mathbf{x}$ is a robust solution, it is not robust to its degree of consensus. To address this issue, in the next section, we propose a new measure called *robust consensus* and discuss how to incorporate it in robust optimization to find robust solutions that are also robust to their degree of consensus.

4.3.4 Robust Consensus and Problem Reformulations

For a solution $\mathbf{x} \in \mathcal{V}$, we define *robust consensus*, denoted by $\mathcal{C}^{\mathcal{R}}(\mathbf{x})$, as a measure of its robustness to its degree of consensus as follows:

$$\mathcal{C}^{\mathcal{R}}(\mathbf{x}) = \frac{1}{|\mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})|} \int_{\mathbf{y} \in \mathcal{B}_\delta^{\mathcal{V}}(\mathbf{x})} \mathcal{C}(\mathbf{y}) d\mathbf{y}. \tag{4.3.9}$$

Here, $\mathcal{C}(\cdot)$ is chosen either as $\mathcal{C}^1(\cdot)$ or $\mathcal{C}^2(\cdot)$. Note that, for any $\mathbf{x} \in \mathcal{V}$, $\mathcal{C}^{\mathcal{R}}(\mathbf{x})|_{\delta=\mathbf{0}} = \mathcal{C}(\mathbf{x})$. In other words, our definitions of robust consensus reduce to their corresponding definitions of consensus when $\delta = \mathbf{0}$. Though at a glance one may find similarities between (1.1.7) and (4.3.9), there are significant differences between them. In (1.1.7),

$\mathcal{G}(\mathbf{y})$ is a "dominance function". It is 1 if $\mathbf{y}$ is a non-dominated solution, and 0, otherwise. Consequently, $\mathcal{G}(\mathbf{y})$ has no relationship with the degree of satisfaction of the DMs for $\mathbf{y}$. On the contrary, $\mathcal{C}(\mathbf{y})$ in (4.3.9) provides a real value in the range $[0, 1]$ that measures the degree of satisfaction of the DMs for $\mathbf{y}$, and hence, $\mathcal{C}(\mathbf{y})$ has nothing to do with the nature of $\mathbf{y}$ in terms of dominance.

Now, we use an expectation based approach to define a robust solution which is also robust with respect to its degree of consensus as follows. A solution $\mathbf{x} \in \mathcal{V}$ (i) is a robust solution, (ii) has consensus, and (iii) is robust with respect to its degree of consensus, if $\mathbf{x}$ is in the Pareto set (PS) of the following minimization problem:

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathcal{V}}{\text{minimize}}\, \mathbf{f}^{e,\mathcal{C}^{\mathcal{R}}}(\mathbf{x}) = \left(f_1^e(\mathbf{x}), f_2^e(\mathbf{x}), \cdots, f_m^e(\mathbf{x}), -\mathcal{C}^{\mathcal{R}}(\mathbf{x})\right);\\
&\text{subject to}\\
&g_j(\mathbf{x}) \leq 0, j = 1, 2, \cdots, n_{\neq};\\
&h_k(\mathbf{x}) = 0, k = 1, 2, \cdots, n_{=}.
\end{aligned} \tag{4.3.10}$$

Next, we use a variance based approach as follows. A solution $\mathbf{x} \in \mathcal{V}$ (i) is a robust solution, (ii) has consensus, and (iii) is robust to its degree of consensus, if $\mathbf{x}$ is in the PS of the following minimization problem:

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathcal{V}}{\text{minimize}}\, \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x}));\\
&\text{subject to}\\
&g_j(\mathbf{x}) \leq 0, j = 1, 2, \cdots, n_{\neq};\\
&h_k(\mathbf{x}) = 0, k = 1, 2, \cdots, n_{=};\\
&\frac{||\mathbf{f}^e(\mathbf{x}) - \mathbf{f}(\mathbf{x})||_p}{||\mathbf{f}(\mathbf{x})||_p} \leq \eta;\\
&\frac{||\mathcal{C}^{\mathcal{R}}(\mathbf{x}) - \mathcal{C}(\mathbf{x})||_p}{||\mathcal{C}(\mathbf{x})||_p} \leq \eta_{\mathcal{C}^{\mathcal{R}}};
\end{aligned} \tag{4.3.11}$$

Here, η and $\eta_{\mathcal{C}^{\mathcal{R}}}$ are two limiting or tolerance parameters. In this investigation, we consider only formulation (4.3.10).

4.4 Experiments and Results

4.4.1 Test Problems and Experimental Framework

In this work, we consider the following test problem, introduced in [63].

$$
\begin{aligned}
&\text{minimize } f_1(\mathbf{x}) = x_1;\\
&\text{minimize } f_2(\mathbf{x}) = H(x_1) + G(\mathbf{x}) \cdot S(x_1);\\
&\text{subject to } 0 \leq x_1 \leq 1;\ -1 \leq x_j \leq 1,\ j = 2, 3, \cdots, n;\\
&\quad \text{where}\\
&\qquad H(x_1) = 1 - x_1^2;\\
&\qquad G(\mathbf{x}) = \sum_{j=2}^{n} \left(10 + x_j^2 - 10\cos\left(4\pi x_j\right)\right);\\
&\qquad S(x_1) = \frac{1}{0.2 + x_1} + x_1^2.
\end{aligned} \tag{4.4.1}
$$

The PS of this problem comprised of all points where $x_j = 0,\ j = 2, 3, \cdots, n$, and $x_1 \in [0, 1]$. Note that, for all of these solutions $S(x_1) = 0$ and $f_2(\mathbf{x}) = 1 - f_1^2(\mathbf{x})$.

We assume that for a given set of parameters $\boldsymbol{\delta} = (\delta_1, \delta_2, \cdots, \delta_n)$, the j^{th} variable of a solution, i.e., x_j is perturbed in the neighbourhood $\left[x_j - \delta_j, x_j + \delta_j\right]$. Consequently, the neighbourhood $\mathcal{B}_{\delta}^{\mathcal{V}}(\cdot)$ of a solution is the n-orthotope with vertices $(x_1 \pm \delta_1, x_2 \pm \delta_2, \cdots, x_n \pm \delta_n)$. Then, the mean effective objective functions $\left(f_j^e(\cdot),\ j = 1, 2,\ \cdots, m\right)$ for a Pareto-optimal solution $\mathbf{x}$, are given as follows [63]:

$$
\begin{aligned}
f_1^e(\mathbf{x}) =& x_1;\\
f_2^e(\mathbf{x}) =& \left(1 - x_1^2\right) - \frac{\delta_1^2}{3} + \left(\frac{1}{2\delta_1}\log\left(\frac{0.2 + x_1 + \delta_1}{0.2 + x_1 - \delta_1}\right) + \right.\\
&\left.\left(x_1^2 + \frac{\delta_1^2}{3}\right)\right) \cdot \left(\sum_{j=2}^{n}\left(10 + \frac{\delta_j^2}{3} - \frac{10}{4\pi\delta_j}\sin\left(4\pi\delta_j\right)\right)\right).
\end{aligned} \tag{4.4.2}
$$

Thus, using (4.4.2), for a given $\boldsymbol{\delta}$, one can theoretically find the robust PF of type I [63].

To approximate the mean objective functions, we randomly generate H points in the $\boldsymbol{\delta}$-neighborhood $\left(\mathcal{B}_{\delta}^{\mathcal{V}}(\cdot)\right)$ of a point $\mathbf{x}$. Throughout this work, we consider $H = 1000$. Moreover, we consider 5 variables, i.e., $n = 5$. Furthermore, unless specified explicitly, we use $\boldsymbol{\delta}_1 = (0.007, 0.014, 0.014, 0.014, 0.014)$, which is one of the perturbations that produces a robust Pareto front (PF) separated by a gap from the original

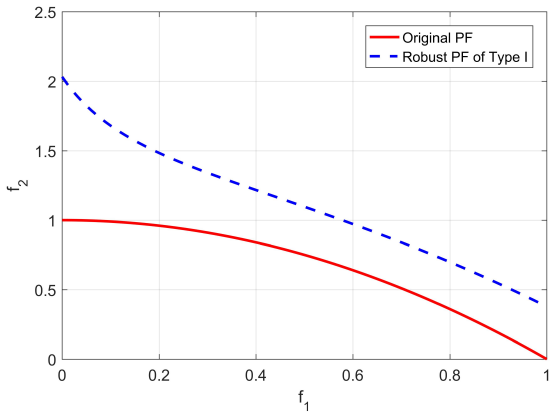


Figure 4.2: Original and type I robust Pareto front of the test problem with $\delta = \delta_1 = (0.007, 0.014, 0.014, 0.014, 0.014)$.

PF [63]. Figure 4.2 shows the original PF and the robust PFs of Type I with $\delta = \delta_1$ for the test problem.

For experimentation, we have used archive-based steady-state micro-genetic algorithm (ASMiGA) [113] with differential evolution crossover-3 (DE-3) [113, 116], polynomial mutation [167] and the following parameter settings: minimum archive size $N_{min} = 20$, maximum archive size $N_{max} = 100$, selection ratio $S_r = 0.15$. Parameters for DE-3 crossover are, $F = 0.5$ and $CR = 0.1$. The distribution index of polynomial mutation is $\eta_m = 50$, and the probability of mutation is $p_m = 1/n$. Note that, any multiobjective algorithm can be used instead of ASMiGA. We, however, choose ASMiGA, as it is a newly designed algorithm proposed by us [113, 116]. ASMiGA was experimentally proven to perform better than popular algorithms like NSGA-II [117] and MOEA/D [168], when tested on several standard benchmark problems [113, 116]. For every test, we have executed the algorithm 10 times and have selected (plotted) only the set of non-dominated solutions from the set of obtained solutions.

4.4.2 A Careful Look at Consensus and Robust Consensus

We have given 12 definitions of consensus, and hence, robust consensus (two approaches $\times$ three definitions of $\psi(\cdot)$ $\times$ two definitions of $\phi(\cdot)$). At first, we examine how the definitions of robust consensus work with $\delta = \mathbf{0}$, i.e., when robust consensus reduces to corresponding consensus. For this purpose, we consider three DMs with the same weights, i.e., $\mathbf{w} = (1/3, 1/3, 1/3)$. We assume that the DMs have provided their preferences in terms of objectives. We consider the following two cases.

Case-1: We presume that the DMs have provided their preferences using Gaussian membership functions with preferences $\mathbf{r}_{11}^{\mathcal{O}} = (0.8, 0.033)$, $\mathbf{r}_{12}^{\mathcal{O}} = (0.9, 0.033)$, $\mathbf{r}_{21}^{\mathcal{O}} = (0.8, 0.0167)$, $\mathbf{r}_{22}^{\mathcal{O}} = (0.9, 0.0167)$, $\mathbf{r}_{31}^{\mathcal{O}} = (0.9, 0.033)$, $\mathbf{r}_{32}^{\mathcal{O}} = (0.8, 0.033)$.

Case-2: We consider that the DMs have provided their preferences using triangular membership functions with preferences $\mathbf{r}_{11}^{\mathcal{O}} = (0.80, 0.10)$, $\mathbf{r}_{12}^{\mathcal{O}} = (0.90, 0.10)$, $\mathbf{r}_{21}^{\mathcal{O}} = (0.80, 0.05)$, $\mathbf{r}_{22}^{\mathcal{O}} = (0.90, 0.05)$, $\mathbf{r}_{31}^{\mathcal{O}} = (0.90, 0.10)$, $\mathbf{r}_{32}^{\mathcal{O}} = (0.80, 0.10)$.

A close look at the parameters reveals that in both cases, their preferences are similar. The preferences are quite specific (the non-specificity of each fuzzy set is low). The prominent difference is that, when Gaussian membership functions are used, every solution would have a nonzero (may be very small) membership value with respect to a given DM's preference. On the contrary, when triangular membership functions are used, the membership values of the solutions lying outside a given triangle would be zero. We want to examine how our formulations behave when (i) each solution has some nonzero membership value (may be very small), and (ii) a set of solutions has zero membership values. For this purpose, we have considered the above two cases.

Figure 4.7 shows the contour plots of solutions for Case-1, when Gaussian membership functions are used for preferences. The top two rows in Fig. 4.7 have six panels one for each of the six combinations of $\phi(\cdot)$ and $\psi(\cdot)$, when Approach I is used. The last two rows, on the other hand, correspond to the same six choices of $\phi(\cdot)$ and $\psi(\cdot)$ but with Approach II. Figure 4.8 depicts the same results as in Fig. 4.7 but using surface plots. To show the effect of choice of membership functions, in Fig. 4.9 we display the contour plots with triangular membership functions (Case-2) for the same problem as in Fig. 4.7.

Figures 4.7, 4.8, and 4.9 reveal that each of the formulations is quite different as their contours and surfaces differ from each other. From the same set of figures, we observe that, when Gaussian membership functions are used, the obtained solutions are spread over a larger region, whereas, for triangular membership functions, the obtained solutions are concentrated over a smaller region. Though detailed results are provided in Figs 4.7, 4.8, and 4.9, to illustrate this visually, we have provided contour plots of consensus and the obtained solutions in Figs. 4.3 (Gaussian membership) and 4.4 (triangular membership) using Approach II with $\phi(\cdot) = \min(\cdot)$. We see that for Gaussian membership functions, the solutions are spread over a larger region and many of them with a low degree of consensus. The reason is that for every solution, a Gaussian membership function provides a nonzero value. However, a triangular membership function assigns a nonzero value for solutions from a specified

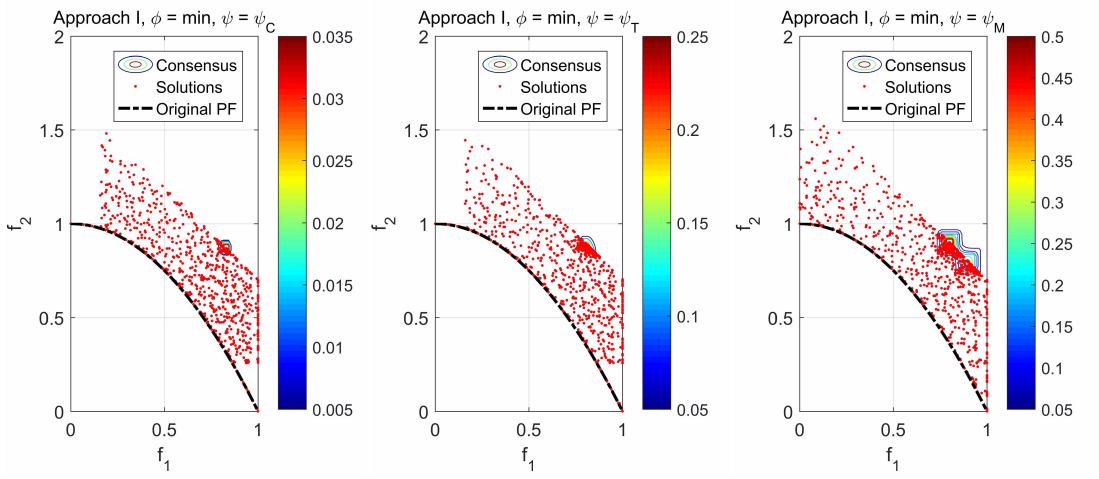


Figure 4.3: Contour plots of different definitions of consensus and the corresponding obtained solutions using Approach I with $\boldsymbol{\delta} = \mathbf{0}$ considering $\phi(\cdot) = \min(\cdot)$, when Gaussian membership functions are used to denote the DMs' preferences.

region and zero value for the rest of the solutions. This makes triangular membership functions more suitable (compared to Gaussian membership functions) when small deviation in the consensus is more important than obtaining a diverse set of solutions. On the other hand, one may prefer Gaussian membership functions over triangular membership functions when the preferences of the DMs are non-overlapping.

From Figs. 4.3 and 4.4, we observe that, due to the addition of consensus as the third objective, the search process ends up with a set of solutions that are crowded over the region where the consensus has a higher value. However, this does not affect the search process to find the Pareto front. Figures 4.7, 4.8, and 4.9 illustrate the same.

Earlier we have discussed that, in some cases Approach I and Approach II may lead to the same definition. However, in many cases, the produced definitions are different, and hence, the spread of the obtained solutions using them, may be visually distinct from each other. Figure 4.5 and the rightmost sub-figure of Fig. 4.3 demonstrate this with an example. In both cases preferences are modeled using Gaussian membership functions. The right most sub-figure of Fig. 4.3 shows the contour of consensus and the obtained solutions using Approach I considering $\phi(\cdot) = \min(\cdot)$ and $\psi(\cdot) = \psi_M(\cdot)$, while Fig. 4.5 depicts the contour of consensus and the obtained solutions using the same $\phi(\cdot)$ and $\psi(\cdot)$ but with Approach II. They clearly reveal the differences in the contours of the corresponding consensus and the obtained solutions. Similar observations can also be made from Figs. 4.7, 4.8, and 4.9.

Now we investigate the behaviour of robust consensus with a nonzero δ, for which

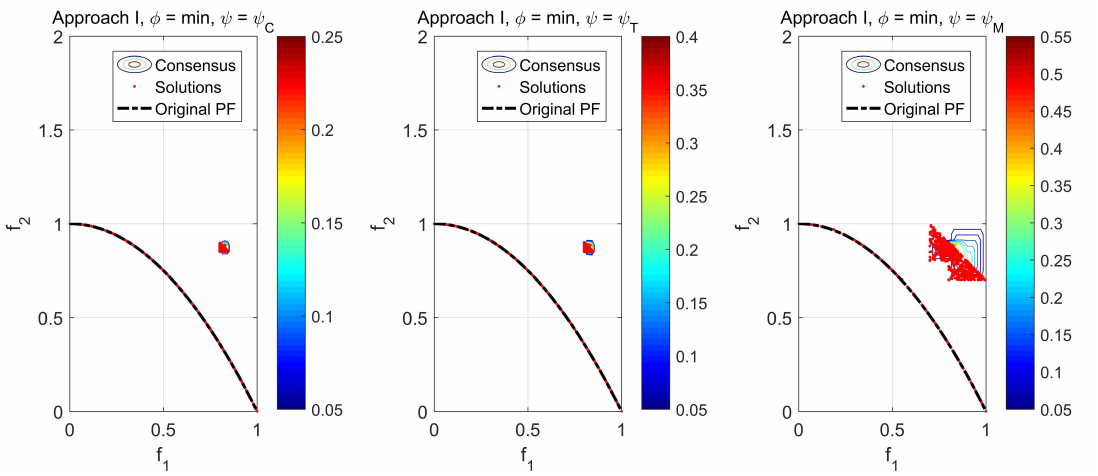


Figure 4.4: Contour plots of different definitions of consensus and the corresponding obtained solutions using Approach I with $\delta = \mathbf{0}$ considering $\phi(\cdot) = \min(\cdot)$, when triangular membership functions are used to denote the DMs' preferences.

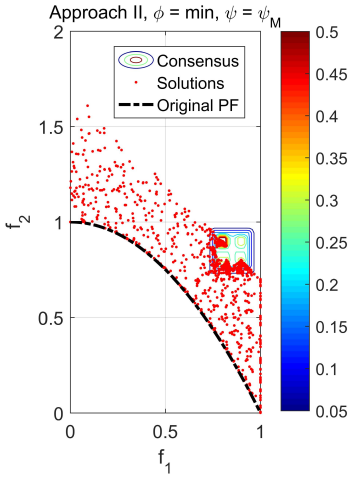


Figure 4.5: Contour plot of consensus and the corresponding obtained solutions using Approach II with $\delta = \mathbf{0}$ considering $\phi(\cdot) = \min(\cdot)$ and $\psi(\cdot) = \psi_M(\cdot)$, when Gaussian membership functions are used to denote the DMs' preferences.

we choose $\delta = \delta_1$. Figs. 4.10 and 4.11 depict the results of this experiment considering the same weights and the same preferences of the DMs as used to generate Figs. 4.7, 4.8, and 4.9. Specifically, Fig. 4.10 illustrates the contours of consensus, when Gaussian membership functions are used. Similarly, Fig. 4.11 shows the same when triangular membership functions are used. Note that, Figs. 4.10 and 4.11 have the same organization as that of Figs. 4.7, 4.8, and 4.9 with respect to their definitions of

consensus.

Though Figs. 4.10 and 4.11 show detailed results of robust-consensus, we consider Figs. 4.5 and 4.6 to discuss some observations regarding robust-consensus. Figs. 4.5 and 4.6 show the results obtained using Approach II, considering $\phi(\cdot) = \min(\cdot)$ and $\psi(\cdot) = \psi_M(\cdot)$, with $\boldsymbol{\delta} = \mathbf{0}$ and $\boldsymbol{\delta} = \boldsymbol{\delta}_1$, respectively, when the preferences are provided using Gaussian membership functions. Consequently, together these two figures illustrate how a perturbation in the system changes the obtained solutions. As expected, unlike Fig. 4.5, in Fig. 4.6 there is no solution in the region bounded by the PF and the robust PF. In addition to this, in Fig. 4.6 there is a crowding of obtained solutions below the region where consensus has a high value. Note that, the contour of consensus and robust consensus are not the same. All figures show the contours/surfaces of consensus. Therefore, when there is a perturbation in the system, i.e., $\boldsymbol{\delta} \neq \mathbf{0}$, we may not get the crowding of solutions where consensus has a peak. However, intuitively, the contours of consensus and robust consensus should be somewhat similar (depending on the nature of the objective function, they could be significantly different also), and hence, with a nonzero $\boldsymbol{\delta}$ we should expect solutions near the region where consensus has a peak. Consequently, in Fig. 4.6 the crowding of non-robust solutions below the peak, where consensus has a peak, is consistent with our intuition. Similar observations can be made from Figs. 4.10 and 4.11. Another observation from these figures is that, if we use triangular membership functions, we do not obtain any solution (or obtain a few solutions) within the region between the PF (robust PF) and the region where the value of consensus is high. It is noteworthy that each of the formulations finds solutions throughout the robust PF.

4.4.3 Effect of Specificity on Robust Consensus

We want to examine how the changes in the specificity of the preferences (provided by the DMs) affect robust consensus. For this purpose, we assume that there are two DMs with weights $\mathbf{w} = (0.5, 0.5)$ and they have provided their preferences in the objective space using Gaussian membership functions. We consider the following three cases of preferences:

Case-1: $\mathbf{r}_{11}^{\mathcal{O}} = (0.8, 0.0167)$, $\mathbf{r}_{12}^{\mathcal{O}} = (0.9, 0.0167)$, $\mathbf{r}_{21}^{\mathcal{O}} = (0.9, 0.033)$, $\mathbf{r}_{22}^{\mathcal{O}} = (0.8, 0.033)$.

Case-2: $\mathbf{r}_{11}^{\mathcal{O}} = (0.8, 0.033)$, $\mathbf{r}_{12}^{\mathcal{O}} = (0.9, 0.033)$, $\mathbf{r}_{21}^{\mathcal{O}} = (0.9, 0.033)$, $\mathbf{r}_{22}^{\mathcal{O}} = (0.8, 0.033)$.

Case-3: $\mathbf{r}_{11}^{\mathcal{O}} = (0.8, 0.033)$, $\mathbf{r}_{12}^{\mathcal{O}} = (0.9, 0.033)$, $\mathbf{r}_{21}^{\mathcal{O}} = (0.9, 0.0167)$, $\mathbf{r}_{22}^{\mathcal{O}} = (0.8, 0.0167)$.

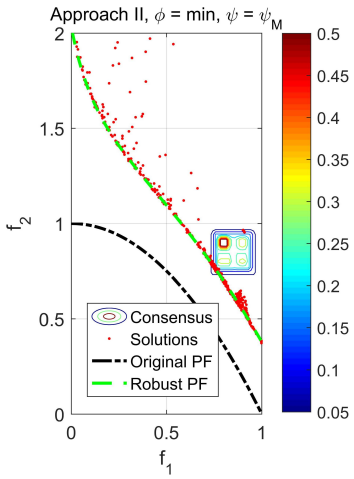


Figure 4.6: Contour plot of consensus and the corresponding obtained solutions using Approach II with $\delta = \delta_1$ considering $\phi(\cdot) = \min(\cdot)$ and $\psi(\cdot) = \psi_M(\cdot)$, when Gaussian membership functions are used to denote the DMs' preferences.

Here, in Case-1, the preferences of the first DM is more specific (higher specificity) than that of the second DM. In Case-2, both DMs are the same in terms of the specificity of preference. In Case-3, the preferences of the second DM have more specificity than that of the first DM. In this experiment, we consider $\phi(\cdot) = \min(\cdot)$. Figure 4.12 shows the contours of consensus along with the obtained solutions with $\delta = \mathbf{0}$. Similarly, Fig. 4.13 shows the same with $\delta = \delta_1$. Figures 4.12 and 4.13 have two columns each with three subfigures one for each of these three cases. The left column corresponds to Approach I and the right column corresponds to Approach II.

The contour plots in Figs. 4.12 and 4.13 reveal that the changes in consensus with changes in the specificities of the preferences (provided by the DMs) are in accordance with what they should be intuitively. To be more specific, the peaks (locations where consensus attains the maximum) in Figs. 4.12 and 4.13 shift from top-left to bottom-right as the specificities of the preferences provided by the first DM decrease and the specificities of the preferences provided by the second DM increase. Moreover, in Fig. 4.12, we observe that a cluster of solutions is found where the values of consensus exhibit a peak. However, from Fig. 4.13, we observe two clusters: one is below the peak and the other one is above the peak. This is probably caused by the nature of the test problem and the perturbations during the evaluations of the solutions.

4.4.4 Effects of Weights on Consensus

To inspect how the changes in the weights of the DMs affect robust consensus, we consider a scenario with two DMs. We assume that the DMs provide their preferences in the objective space using Gaussian membership function with parameters $\mathbf{r}_{11}^{\mathcal{O}} = (0.8, 0.033)$, $\mathbf{r}_{12}^{\mathcal{O}} = (0.9, 0.033)$, $\mathbf{r}_{21}^{\mathcal{O}} = (0.9, 0.033)$, $\mathbf{r}_{22}^{\mathcal{O}} = (0.8, 0.033)$. Here, we consider three cases of weights: (i) $\mathbf{w} = (0.1, 0.9)$, (ii) $\mathbf{w} = (0.5, 0.5)$, and (iii) $\mathbf{w} = (0.9, 0.1)$. In this experiment, we consider $\phi(\cdot) = \min(\cdot)$ and $\delta = \delta_1$. In Fig. 4.14 we display the results of these experiments using contour plots. Fig. 4.14 has two columns, each with three subfigures, one for each of these weights. The left column corresponds to Approach I and the right column corresponds to Approach II.

Figure 4.14 reveals that the changes in consensus with the changes in the weights of the DMs are in accordance with our intuition. Specifically, for the first approach, the peaks in Fig. 4.14 shift from bottom-right to top left as the weight of the first DM increases and the weight of the second DM decreases. Similarly, for the second approach, the common region of interest of the DMs, in Fig. 4.14, is shifted from bottom-right to top left with the same changes in weights. We also observe that the contour plots of these two approaches and the changes in the plots are quite different from each other.

4.5 Conclusions and Discussions

We have proposed a framework to define *consensus* to measure the level of mutual agreement among a set of DMs for a given FGDM-MOP. This framework can be used to generate many definitions of consensus. Then, we have defined another indicator, called *robust consensus*, to measure the robustness of a solution to its degree of consensus. After that, we have proposed two ways to reformulate a given MOP problem so that solutions of the reformulated problem are robust to their degree of consensus. Lastly, we have investigated the behaviour of these definitions and reformulations when the preferences are provided in the objective space. We have also investigated the changes in the nature of solutions when the specificities of the preferences provided by the DMs change. The effect of the changes in the weights or the importance of the DMs has also been studied. We have used contour plots and surface plots to depict the results of our investigations.

From our limited investigation we found that, though for some choices of aggregation operators Approach I and Approach II lead to the same definition, in many cases, the definitions are notably different. Moreover, the choice of membership functions

also has a significant impact on the degree of consensus and can make the outcomes differ significantly. The choice of membership functions also depends on the problems/DMs. If one wants to get a diverse set of solutions sacrificing the degree of consensus, Gaussian membership functions would be preferred over triangular membership functions. On the other hand, if we prefer to get a small set of solutions with a higher degree of consensus, then triangular membership functions may be preferred. We note here that irrespective of the choices of different components, every formulation ends up finding solutions throughout the (robust) PF. We also have observed that the effect of specificity on robust consensus and the effect of weights on consensus are consistent with our intuitions. Lastly, we note here that one may prefer Approach II over Approach I if she wants the weights to play a stronger role. Further, one can control the nature of the influence of the weights to some extent by choosing a suitable $\psi(\cdot)$. Finally, we mention that we could not compare our work with any existing work, because, we could not find any work that deals with "robust consensus"

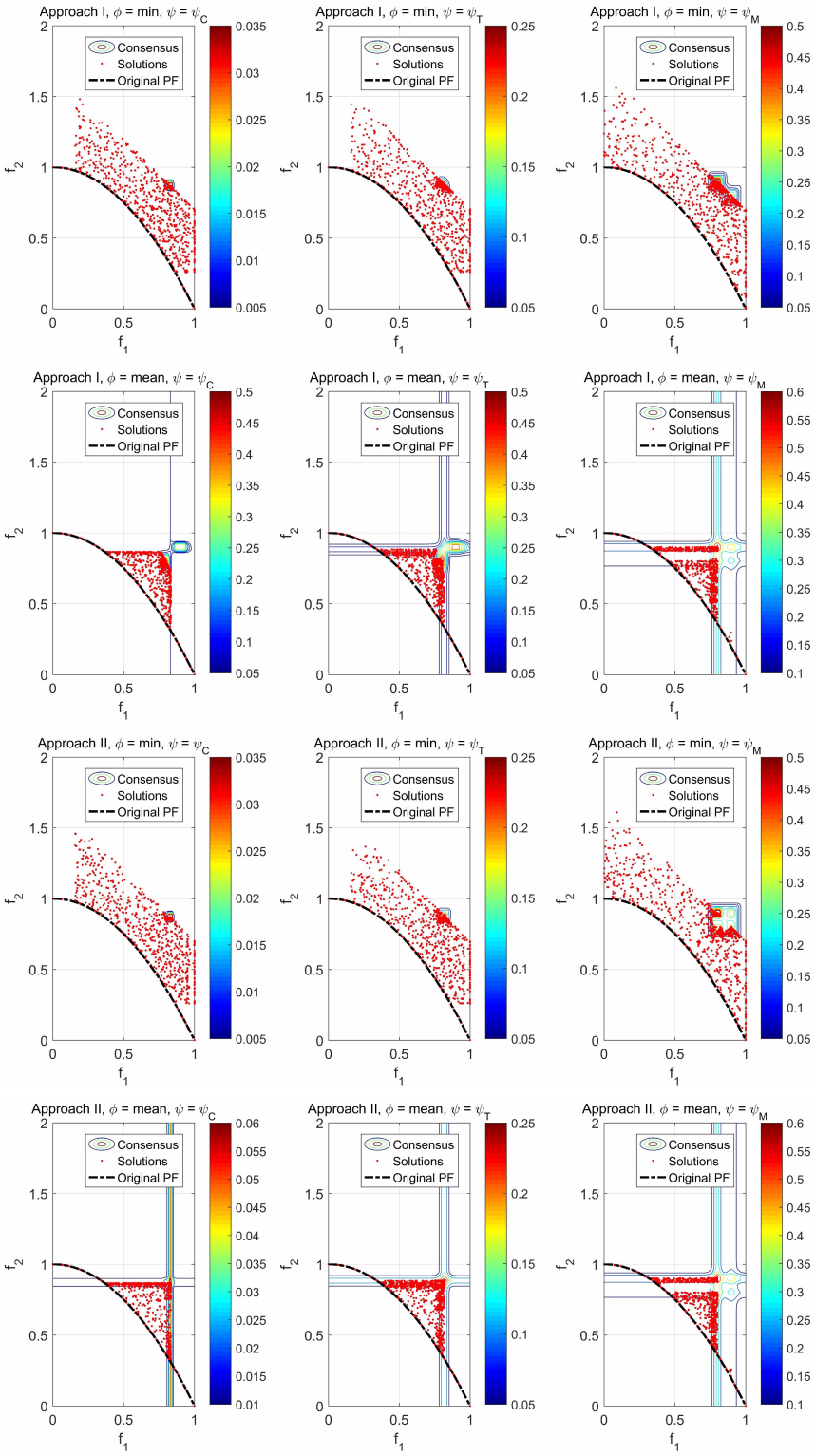


Figure 4.7: Contour plots of different definitions of consensus and the corresponding obtained solutions with $\delta = \mathbf{0}$ when Gaussian membership functions are used to denote the DMs' preferences.

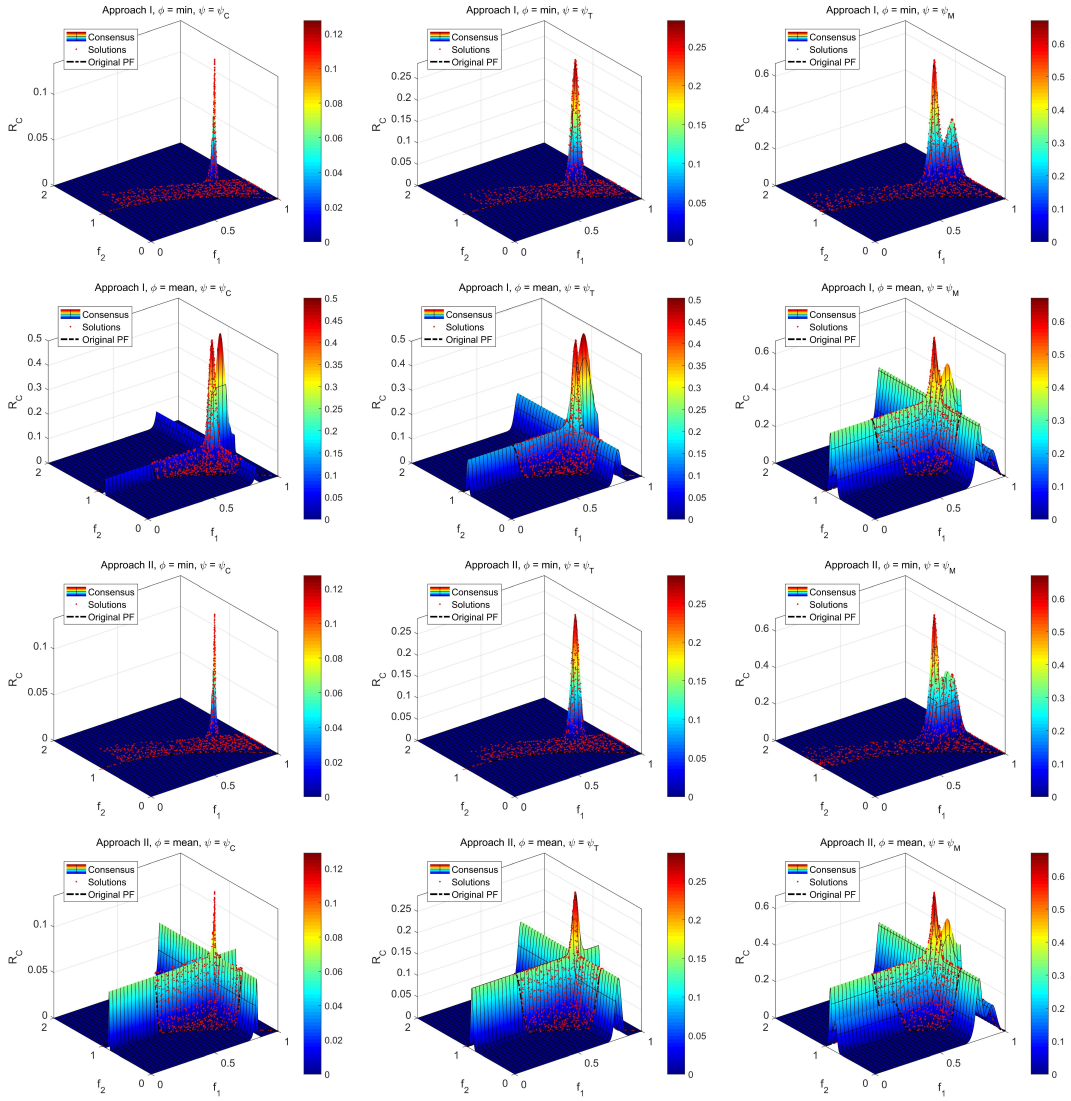


Figure 4.8: Surface plots of different definitions of consensus and the corresponding obtained solutions with $\delta = \mathbf{0}$ when Gaussian membership functions are used to denote the DMs' preferences.

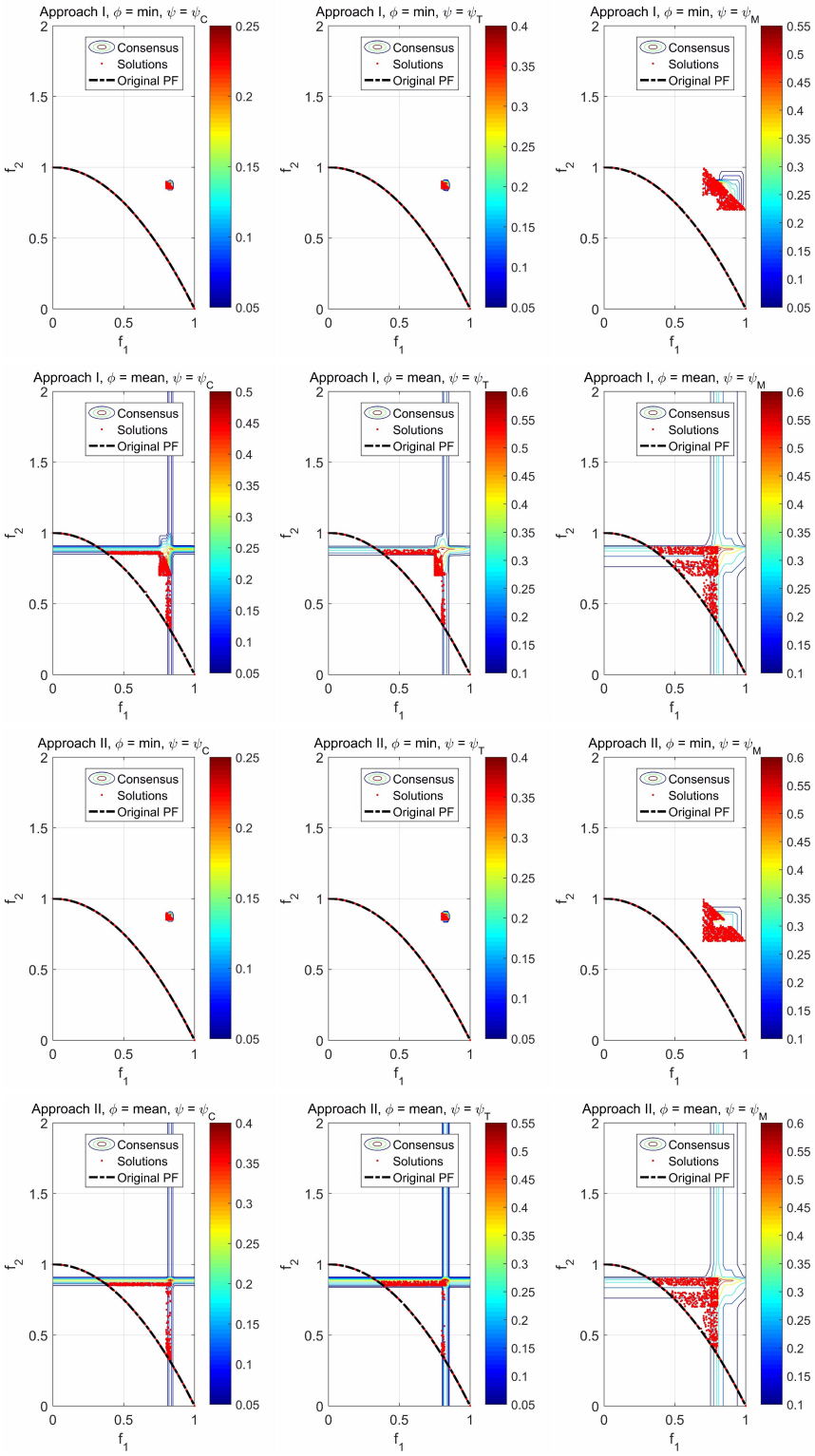


Figure 4.9: Contour plots of different definitions of consensus and the corresponding obtained solutions with $\boldsymbol{\delta} = \mathbf{0}$ when triangular membership functions are used to denote the DMs' preferences.

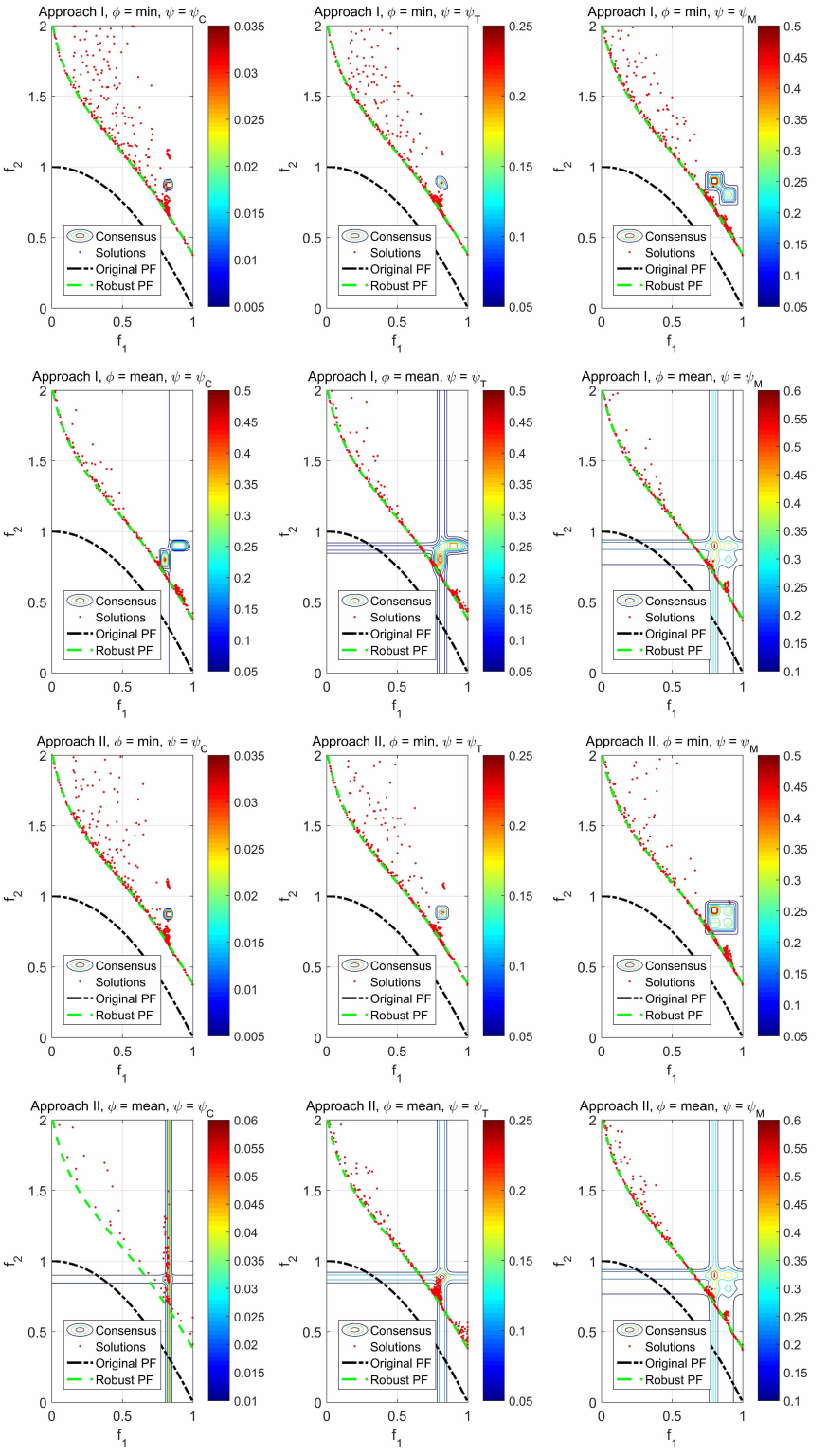


Figure 4.10: Contour plots of different definitions of consensus and the corresponding obtained solutions with $\boldsymbol{\delta} = \boldsymbol{\delta}_1 = (0.007, 0.014, 0.014, 0.014, 0.014)$ when Gaussian membership functions are used to denote the DMs' preferences.

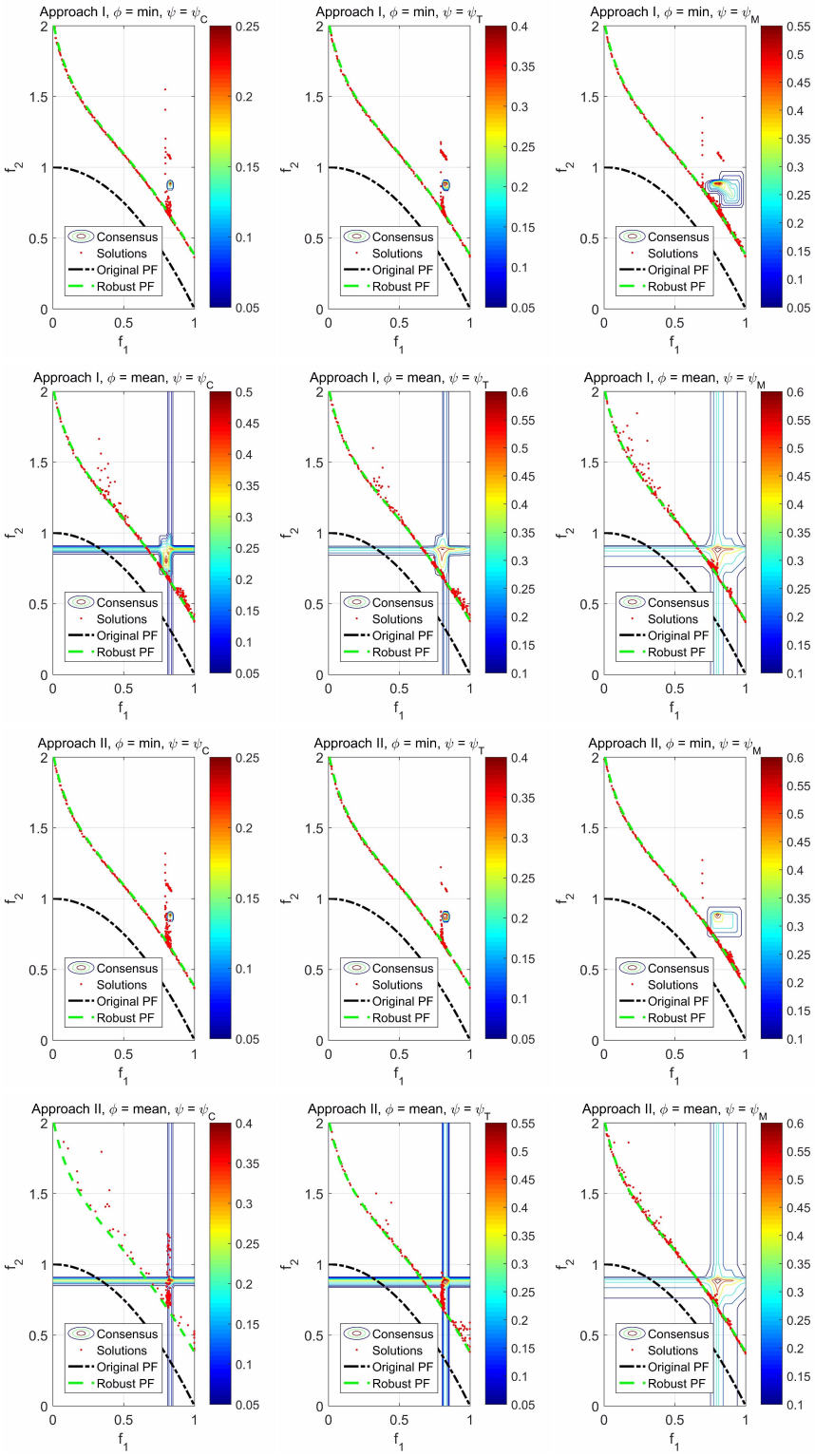


Figure 4.11: Contour plots of different definitions of consensus and the corresponding obtained solutions with $\delta = \delta_1 = (0.007, 0.014, 0.014, 0.014, 0.014)$ when triangular membership functions are used to denote the DMs' preferences.

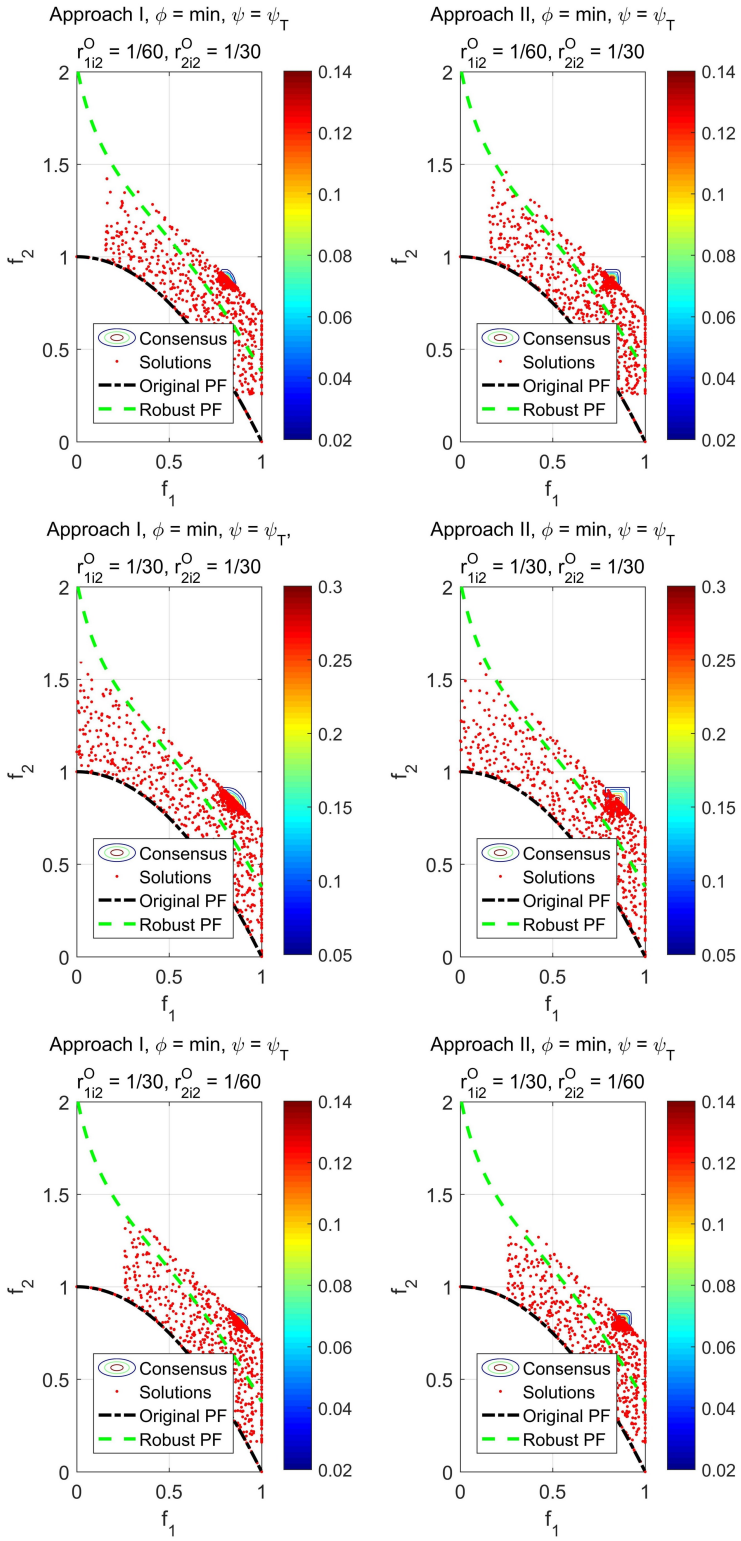


Figure 4.12: Contour plots of different definitions of consensus and the obtained solutions considering different specificities of the preferences (provided by the decision makers) with $\boldsymbol{\delta} = \mathbf{0}$ and $\phi(\cdot) = \min(\cdot)$. Preferences are provided using Gaussian membership functions.

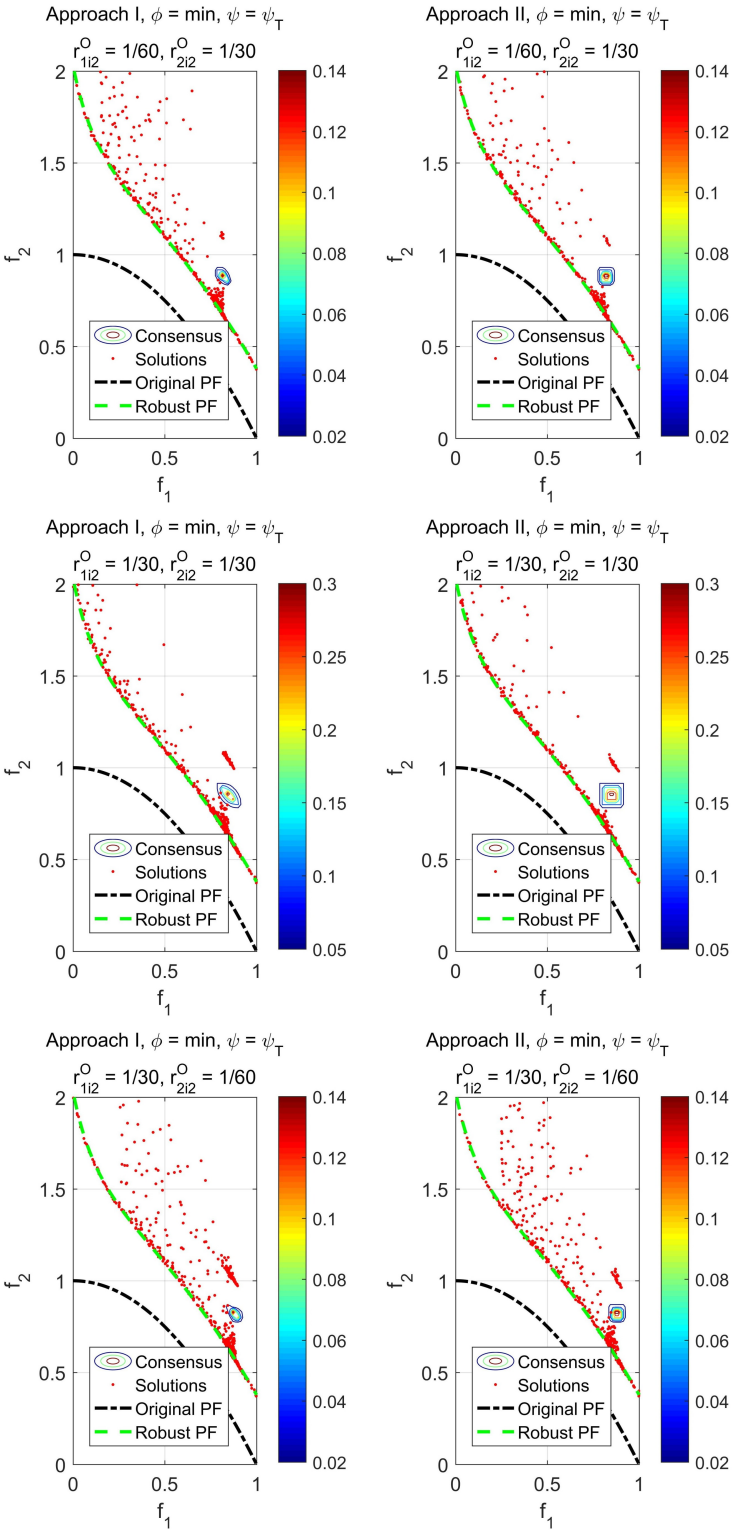


Figure 4.13: Contour plots of different definitions of consensus and the obtained solutions considering different specificities of the preferences (provided by the decision makers) with $\delta = \delta_1 = (0.007, 0.014, 0.014,\ 0.014, 0.014)$ and $\phi(\cdot) = \min(\cdot)$. Preferences are provided using Gaussian membership functions.

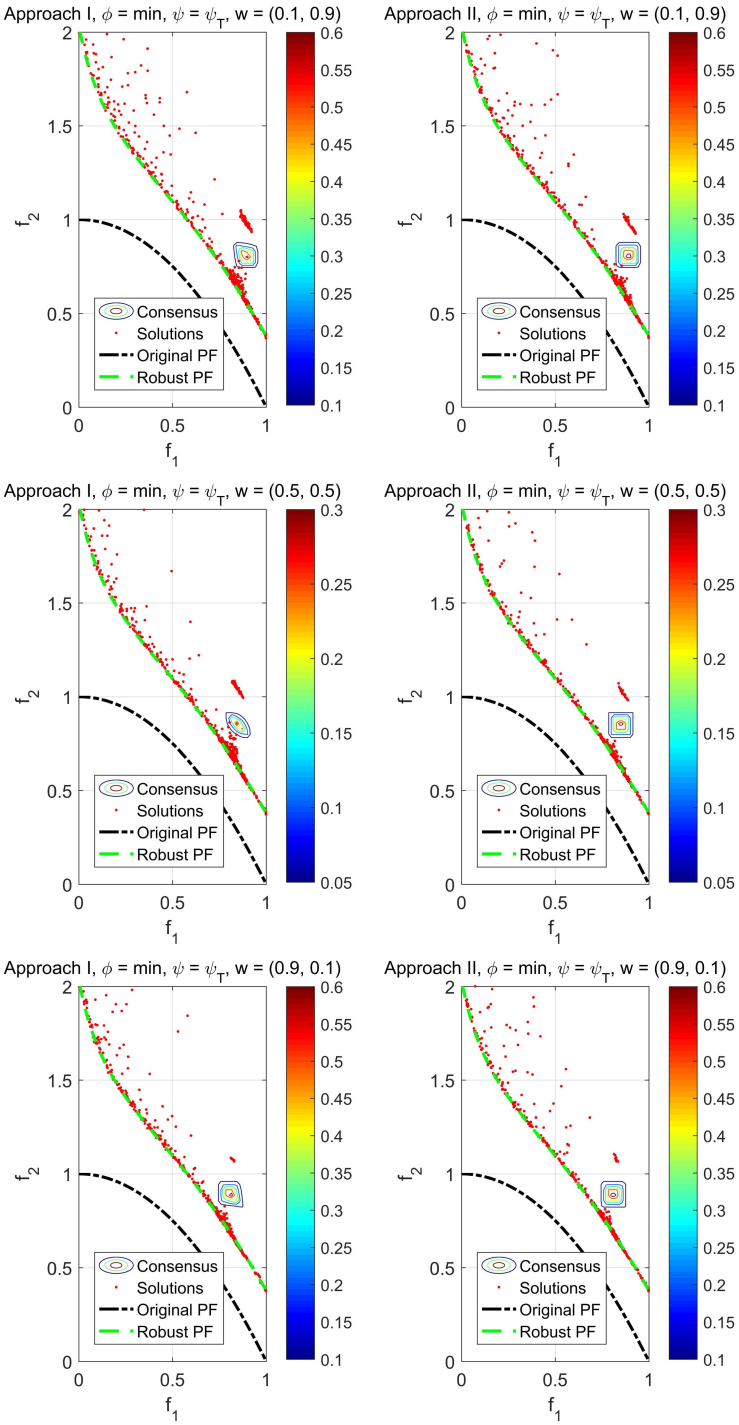


Figure 4.14: Contour plots of different definitions of consensus and the obtained solutions considering different weights associated with the decision makers. $\delta = \delta_1 = (0.007, 0.014, 0.014, 0.014, 0.014)$ and $\phi(\cdot) = \min(\cdot)$ are used. Preferences are provided using Gaussian membership functions.

Chapter 5

Robustness in Multiobjective Evolutionary Optimization [5]

5.1 Introduction

The concept of robustness in multiobjective optimization problems (MOPs) is largely similar to robustness in single objective optimization problems [60]. Despite that, the number of attempts made for uncertainty handling in MOPs is notably smaller than that for uncertainty handling in single objective optimization problems [60]. Most likely this is because in case of MOPs, sensitivity needs to be accounted with respect to each of the objectives. In other words, to *measure* sensitivity, a collective effect of deviations in each of the objectives need to be used [63]. Consequently, four possible robust fronts can be found compared to the Pareto front (PF) [63]. First, entire PF is robust. Second, a part of the PF is robust and there are no other robust solutions. Third, a part of the PF is robust and there are other robust solutions. Fourth, no part of the PF is robust, and hence, the entire robust front is composed of a local front or multiple local fronts.

In this Chapter, first, we have discussed some salient issues related to existing methodologies [63] that find robust multiobjective solutions. In that regard, we have demonstrated how these methodologies cannot discriminate the robust solutions in terms of their robustness/sensitivity. We have then proposed a new measure of sensitivity to assess the robustness of multiobjective solutions. The underlying idea has been taken from [169, 170], which have used a similar definition of sensitivity in the context of neural networks and discrete time linear systems. Next, we have shown how the cost of computing sensitivity can be reduced using an approximation subject

to the following conditions. First, the objective functions are differentiable. Second, the input noise is any one of uniform noise, additive white Gaussian noise (AWGN), and multiplicative noise (MN). We have proposed three approaches to reformulate MOPs. The reformulated MOP using the first approach yields solutions of the original MOP with different degrees of sensitivity/robustness. The other two reformulated MOPs yield solutions with lower degrees of sensitivity than a given limit. Using extensive experiments on six test problems, we have validated our claims and have analyzed the behaviours of the reformulation strategies.

5.2 Related Works, Motivation, and Objectives

5.2.1 The Literature

The state-of-the-art on robust optimization in MOPs is sufficiently affluent [57, 60, 62–72, 171, 172]. There are some other works [173, 174] that have reformulated a given single objective optimization problem as an MOP to find robust solutions. In [57] a confidence measure is proposed, which calculates the confidence that we have in terms of robustness on a particular multiobjective solution. We have discussed some of the related works in Section 1.1.5, where we have introduced robust optimization. Here, we discuss some other related works.

In [171, 172] researchers have proposed a way to define the degree of robustness of a multiobjective solution $\mathbf{x} \in \mathcal{V}$. Let, $\mathcal{B}_{\delta}^{\mathcal{V}}(\mathbf{x}) \in \mathcal{V}$ be a hypercubic neighbourhood of $\mathbf{x}$, such that, it is centered at $\mathbf{x}$ and 2δ is the length of its edge. Let us also assume that $\mathcal{B}_{\eta}^{\mathcal{O}}(\mathbf{f}(\mathbf{x})) \in \mathcal{O}$ is a hyperspherical neighbourhood of $\mathbf{f}(\mathbf{x})$, such that, it is centered at $\mathbf{f}(\mathbf{x})$ and η is its radius. Then, the degree of robustness of $\mathbf{x}$ is an integer k, such that, for a given threshold p, the following two conditions hold. First, the percentage of solutions in $\mathcal{B}_{k\delta}^{\mathcal{V}}(\mathbf{x})$, whose objective function values belong to $\mathcal{B}_{\eta}^{\mathcal{O}}(\mathbf{f}(\mathbf{x}))$, is greater than or equal to p. Second, the percentage of solutions in $\mathcal{B}_{(k+1)\delta}^{\mathcal{V}}(\mathbf{x})$, whose objective function values belong to $\mathcal{B}_{\eta}^{\mathcal{O}}(\mathbf{f}(\mathbf{x}))$, is lower than p.

In [173], the authors have proposed two robustness measures in the context of single objective optimization problems, which may appear somewhat similar to the measure proposed in this work but in reality they are significantly different. Let there be a set of solutions (a population) $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$; $\mathbf{x}_i \in \mathcal{V} \subseteq \mathbb{R}^n$, $i = 1, 2, \cdots, N$; with N solutions associated with the objective function $f(\cdot)$. Then, for the j^{th} solution

$\mathbf{x}_j$ the first robust measure [173] is defined as follows:

$$f^{R1}(\mathbf{x}_j) = \frac{1}{n}\sum_{i=1}^{n}\frac{\bar{\sigma}_{f,j}}{\sigma_{x_i}}, \tag{5.2.1}$$

where $\bar{\sigma}^2_{f,j}$ is an estimation of the variance of $f(\cdot)$ in the neighbourhood of $\mathbf{x}_j$ and $\sigma^2_{x_i}$ is the variance of x_i. Here, $\bar{\sigma}^2_{f,j}$ is defined as follows:

$$\bar{\sigma}^2_{f,j} = \frac{1}{N_j}\sum_{\substack{\mathbf{y}\in\mathcal{B}^{\mathcal{V}}_{\delta}(\mathbf{x}_j)\bigcap\mathcal{P}\\ \mathbf{x}\neq\mathbf{y}}}\sum_{i=1}^{n}\left(\frac{f(\mathbf{x}_j)-f(\mathbf{y})}{x_{ji}-y_i}\right)^2. \tag{5.2.2}$$

Here, $\mathcal{B}^{\mathcal{V}}_{\delta}(\mathbf{x}_j) \in \mathcal{V}$ is a hyperspherical neighbourhood of $\mathbf{x}_j$ with radius δ and $N_j = |\mathcal{B}^{\mathcal{V}}_{\delta}(\mathbf{x}_j)\bigcap\mathcal{P}|$. The second robustness measure [173] is defined as follows:

$$f^{R2}(\mathbf{x}_j) = \frac{\sigma_{f,j}}{\bar{\sigma}_{\mathbf{x},j}}, \tag{5.2.3}$$

where $\sigma^2_{f,j}$ is the local variance of $f(\cdot)$ estimated in the neighborhood of $\mathbf{x}_j$ and $\bar{\sigma}_{\mathbf{x},j}$ is the local standard deviation of x_i calculated in the neighbourhood of $\mathbf{x}_j$, which are defined as follows:

$$\sigma^2_{f,j} = \frac{1}{N_j-1}\sum_{\mathbf{y}\in\mathcal{B}^{\mathcal{V}}_{\delta}(\mathbf{x}_j)\bigcap\mathcal{P}}(f(\mathbf{y})-\mu_{f,j})^2, \tag{5.2.4}$$

$$\bar{\sigma}_{\mathbf{x},j} = \frac{1}{n}\sum_{i=1}^{n}\sigma_{x_i,j}. \tag{5.2.5}$$

Here, $\mu_{f,j}$ is the local mean of $f(\cdot)$ and $\sigma^2_{x_i,j}$ is the local variance of x_i, which are estimated in the neighbourhood of $\mathbf{x}_j$. They are defined as follows, respectively.

$$\mu_{f,j} = \frac{1}{N_j}\sum_{\mathbf{y}\in\mathcal{B}^{\mathcal{V}}_{\delta}(\mathbf{x}_j)\bigcap\mathcal{P}}f(\mathbf{y}), \tag{5.2.6}$$

$$\sigma^2_{x_i,j} = \frac{1}{N_j}\sum_{\mathbf{y}\in\mathcal{B}^{\mathcal{V}}_{\delta}(\mathbf{x}_j)\bigcap\mathcal{P}}(y_i-\mu_{x_i,j})^2. \tag{5.2.7}$$

Here, $\mu_{x_i,j}$ is the local mean of x_i (the i^{th} variable) estimated using only the solutions

of $\mathcal{P}$ that are in the neighbourhood of $\mathbf{x}_j$.

$$\mu_{x_i,j} = \frac{1}{N_j} \sum_{\mathbf{y} \in \mathcal{B}_\delta^{\mathcal{Y}}(\mathbf{x}_j) \cap \mathcal{P}} y_i. \tag{5.2.8}$$

5.2.2 Issues with Type I and Type II Robust Solutions: Our Motivation

One of the major works regarding robustness in MOPs is by Deb et. al [63]. In that work, they did not define any measure of robustness/sensitivity of multiobjective solutions. Instead, they defined two types of robust solutions, denoted as type I and type II robust solutions. They provided two ways to reformulate MOPs to obtain these two types of robust solutions. They [63] claimed that all solutions of the reformulated MOP are robust solutions. However, they did not provide any measure to compare these solutions in terms of their robustness/sensitivity. We note here that these solutions may have different degrees of deviations when the same degrees of perturbations are applied in the variable space. To establish our claim, at first, we define a measure, called mean of deviations ($\texttt{MoD}[\cdot,\cdot]$). For a solution $\mathbf{x} \in \mathcal{V}$ and a set of N perturbations $\Delta\mathcal{X} = \{\Delta\mathbf{x}_1, \Delta\mathbf{x}_2, \cdots, \Delta\mathbf{x}_N\}$, it is defined as follows.

$$\texttt{MoD}\left[\mathbf{x}, \Delta\mathcal{X}\right] := \frac{1}{N} \sum_{i=1}^{N} \texttt{d}[\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}_i), \mathbf{f}(\mathbf{x})], \tag{5.2.9}$$

where $\texttt{d}[\cdot,\cdot]$ is a distance measure. In this Chapter, we use $\texttt{d}[\cdot,\cdot]$ as the Euclidean distance. Note that, a solution with a low value of $\texttt{MoD}[\cdot,\cdot]$ has a low degree of sensitivity, and hence, a high degree of robustness. If there is no discrimination among the type I robust solutions of a given problem, their $\texttt{MoD}[\cdot,\cdot]$s should be close. However, our claim is that there may be large differences in terms of $\texttt{MoD}[\cdot,\cdot]$. To empirically validate our claim, we perform the following experiment.

At first we consider Test Problem 1 (TP1), which was introduced in [63] (see the Appendix for a detailed description of TP1). It is a bi-objective test problem (TP). The set of points with $x_i = 0,\ i = 2, 3, \cdots, n$, and $x_1 \in [0, 1]$ comprise the Pareto set (PS) of this problem. Moreover, for each solution $\mathbf{x}$ in the PS, $f_2(\mathbf{x}) = 1 - f_1^2(\mathbf{x})$ hold true. We use the archive-based steady-state micro-genetic algorithm (ASMiGA) [113, 116] as the multiobjective optimizer for this experiment. We choose this one because it is a new algorithm proposed by us and it has been empirically proven to perform better than several state-of-the-art popular algorithms [113, 116]. We solve the type I robust version of the above-mentioned TP for ten times using ASMiGA, and then, we choose the union of the set of obtained solutions. For each run, the operators and parameters

required for this algorithm are as follows: number of function evaluations = 50000, maximum archive size $(N_{max}) = 200$, minimum archive size $(N_{min}) = 100$, selection ratio $(S_r) = 0.15$, differential evolution-3 crossover (with scaling factor $F = 0.5$ and crossover ratio $CR = 0.1$), and polynomial mutation (with $\eta_m = 50.0$). To evaluate the robust objective functions of each solution, we generate 1000 perturbation (noise) vectors using the Latin hypercube (LH) sampling within the hyper-cubic neighborhood with vertices at $(\pm\delta, \pm\delta, \cdots, \pm\delta)^T$, where $\delta = 0.010$. For each evaluation, a different set of perturbations has been used. The obtained solutions and the PF (non-robust) of this TP have been shown in Fig. 5.1. We note here that in all figures in this Chapter and in all figures of the Appendix, there may be some solutions (mostly boundary solutions) outside the plotted area/volume/hypervolume, which we have ignored to keep the figures comprehensible. Now, we compute the $\texttt{MoD}[\cdot,\cdot]$s of these solutions using the same set of perturbations containing 1000 perturbations (generated using the same way as earlier). Here, to be fair, we remove the boundary solutions which go beyond the lower bound or the upper bound of any of the variables after applying any of the perturbations. The mean and standard deviation of the $\texttt{MoD}[\cdot,\cdot]$s of these solutions are 0.2206 and 0.0757, respectively. Figure 5.2 shows the histogram of the $\texttt{MoD}[\cdot,\cdot]$s of these solutions. From these statistics and this figure, we observe that these solutions have notably varying deviations. To investigate further, we select the two solutions with the minimum and the maximum $\texttt{MoD}[\cdot,\cdot]$s. Now, we generate a set of 1000 perturbation vectors using the same way as earlier. Next, we perturb both the solutions using these perturbation vectors and measure the deviations (in terms of Euclidean distance) from the unperturbed solution in the objective space. Thus, we obtain a set of 1000 deviations corresponding to each of the solutions. Figure 5.3 illustrates two histograms corresponding to each of these two set of deviations. From Fig. 5.3, we observe that there is a noticeable difference between these two set of deviations. The solution with the minimum $\texttt{MoD}[\cdot,\cdot]$ indeed is less sensitive, i.e., more robust. Thus, we empirically validate that there are notable differences among the robust solutions of type I in terms of sensitivity. Probably, this is because type I solutions take into account only the central tendency of the perturbed solutions in the objective space and do not consider the deviation of the same. Type II solutions are also affected by a similar issue: for a given solution, the constraint of the problem reformulation takes into account only the distance between the central tendency (or the worst solution generated by the set of perturbations) from the unperturbed point in the objective space and do not consider the deviations of the perturbed points.

Another issue with type I and type II robust solutions is that they do not incor-

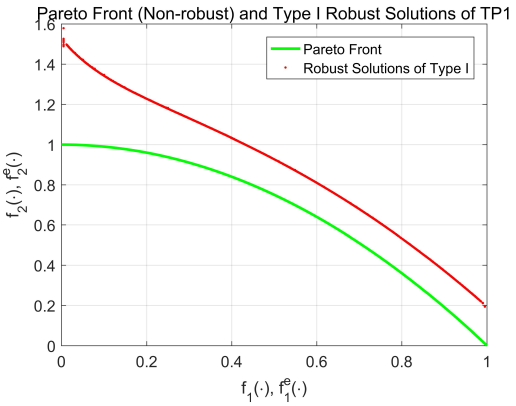


Figure 5.1: Pareto front (non-robust) of TP1 and obtained type I robust solutions on TP1.

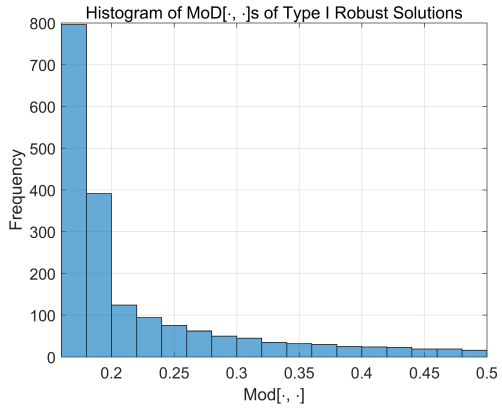


Figure 5.2: Histogram of $\texttt{MoD}[\cdot,\cdot]$s for obtained type I robust solutions.

porate any attribute of the input noise, e.g., the variance of the perturbations vectors. However, a good measure should take into account some properties of the input noise while finding robust solutions or measuring the robustness of a solution.

5.2.3 The Objectives and the Contributions of This Chapter

Here our objectives and contributions in contrast with the issues of the existing methods are as follows.

1. We have demonstrated that type I robust solutions cannot identify the solutions which are more *sensitive* with respect to their variables. Moreover, in the literature there is a lack of measures that quantify the sensitivity/robustness of a solution with respect to perturbations in the variable space. To fill this research gap,

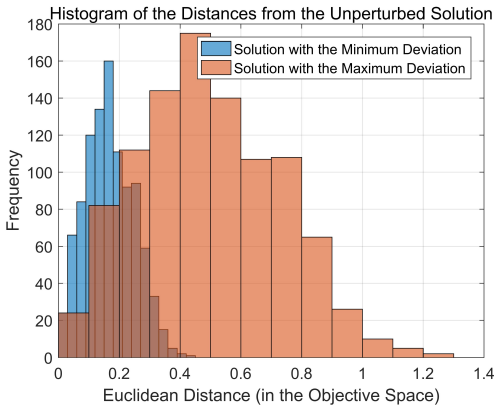


Figure 5.3: Histograms of the distances from the unaltered solutions for the solutions with the minimum and the maximum $\mathtt{MoD}[\cdot,\cdot]$ values.

we propose to use a measure of sensitivity. We also present three approaches to reformulate MOPs. When the first approach is used, the reformulated MOP yields solutions of the original MOP with different degrees of sensitivity. For the other two approaches, the reformulated MOPs yield solutions with sensitivity less than a given threshold.

2. Most methods in the literature assume that the solutions are perturbed inside a hypercube around the solutions with a uniform distribution. However, AWGN or MN in many cases can be a better model of noise for representing real-world perturbations. Our proposed measure and the reformulation methods have been applied to uniform noise as well as to AWGN and MN.

3. Explicit robustness handling in MOPs usually involves generation and evaluation of a large number of solutions in the neighbourhood of a given solution. This demands a huge computational effort, especially when each function evaluation is costly. However, if we consider uniform noise, AWGN, or MN, then for an MOP with differentiable objective functions, when the input perturbations are very small, the proposed measure can be approximated without evaluating a large number of solutions. This may significantly reduce the cost of handling robustness in MOPs.

5.3 Proposed Work

5.3.1 Sensitivity

Let us assume that a solution $\mathbf{x}^* \in \mathcal{V}$ is perturbed by a small input perturbation vector $\Delta\mathbf{x} = (\Delta x_1, \Delta x_2, \cdots, \Delta x_n)^T \in \mathbb{R}^n$, such that, $\texttt{var}[\Delta x_i] = \sigma^2, i = 1, 2, \cdots, n$. Here, $\texttt{var}[\cdot]$ denotes the variance of $[\cdot]$. Due to this perturbation, the deviation in the j^{th} objective can be written as

$$\Delta f_j(\mathbf{x}^*) = f_j(\mathbf{x}^* + \Delta\mathbf{x}) - f_j(\mathbf{x}^*). \tag{5.3.1}$$

Now, following [169,170], we define the sensitivity of the j^{th} objective function $\left(\mathcal{S}_j(\cdot)\right)$ at $\mathbf{x}^*$ as follows.

$$\mathcal{S}_j(\mathbf{x}^*) := \frac{\sqrt{\texttt{var}\left[\Delta f_j(\mathbf{x}^*)\right]}}{\sigma}. \tag{5.3.2}$$

This definition is slightly different from the definitions used in [169,170]. Also note its differences from definitions (5.2.1) and (5.2.3) [173]. Next, following [170], we define the sensitivity $(\mathcal{S}(\cdot))$ at $\mathbf{x}^*$ as follows.

$$\mathcal{S}(\mathbf{x}^*) := \phi\left(\mathcal{S}_1(\mathbf{x}), \mathcal{S}_2(\mathbf{x}), \cdots, \mathcal{S}_m(\mathbf{x})\right), \tag{5.3.3}$$

where $\phi(\cdot)$ is any suitable aggregation operator. In this Chapter, we consider $\phi(\cdot) = \text{mean}(\cdot)$. In some cases, however, $\max(\cdot)$ or some other aggregation operators may be more suitable than $\text{mean}(\cdot)$. Any weighted aggregation operator, e.g., weighted mean operator, can also be used here. Note that, a solution with a high value of $\mathcal{S}(\cdot)$ has a high degree of sensitivity and a low degree of robustness.

Let us assume that the changes in the variables are very small. Now, if $f_j(\mathbf{x}), j = 1, 2, \cdots, m$, is differentiable, then using Taylor series expansion of $f_j(\mathbf{x}^* + \Delta\mathbf{x})$, we get

$$\begin{aligned} f_j(\mathbf{x}^* + \Delta\mathbf{x}) &= f_j(\mathbf{x}^*) + \nabla f_j(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta\mathbf{x} + \cdots \\ &\approx f_j(\mathbf{x}^*) + \nabla f_j(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta\mathbf{x}. \end{aligned} \tag{5.3.4}$$

Here we ignore the higher order terms on the right-hand side of (5.3.4) for very small

changes in variables and $\nabla f_j(\mathbf{x})$ is denoted as

$$\nabla f_j(\mathbf{x}) = \left(\frac{\partial f_j(\mathbf{x})}{\partial x_1}, \frac{\partial f_j(\mathbf{x})}{\partial x_2}, \cdots, \frac{\partial f_j(\mathbf{x})}{\partial x_n} \right)^T. \tag{5.3.5}$$

We use $(\cdot)$ to denote the dot product of two vectors. Next, from (5.3.1) and (5.3.4) we get

$$\begin{aligned} \Delta f_j(\mathbf{x}^*) &\approx f_j(\mathbf{x}^*) + \nabla f_j(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta\mathbf{x} - f_j(\mathbf{x}^*) \\ &= \nabla f_j(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta\mathbf{x}. \end{aligned} \tag{5.3.6}$$

5.3.2 Sensitivity to Additive White Gaussian Noise

If we assume that the noise is an AWGN, i.e., $\Delta x_i \sim \mathcal{N}(0, \sigma^2); i = 1, 2, \cdots, n$; then the following holds.

$$\mathtt{E}[\Delta x_i^2] = \sigma^2; i = 1, 2, \cdots, n; \tag{5.3.7}$$

$$\mathtt{E}[\Delta x_i \Delta x_k] = 0; i = 1, 2, \cdots, n; k \neq i; \tag{5.3.8}$$

$$\begin{aligned} \mathtt{E}[\Delta f_j(\mathbf{x}^*)] &\approx \mathtt{E}[\nabla f_j(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta\mathbf{x}] \\ &= \sum_{i=1}^{n} \left. \frac{\partial f_j(\mathbf{x})}{\partial x_i} \right|_{\mathbf{x}=\mathbf{x}^*} \mathtt{E}\left[\Delta x_i\right] \\ &= 0; j = 1, 2, \cdots, m; \end{aligned} \tag{5.3.9}$$

where $\mathtt{E}[\cdot]$ denotes the expectation of $[\cdot]$. Next, we express $\mathtt{var}[\Delta f_j(\mathbf{x}^*)]$ as

$$\begin{aligned} \mathtt{var}[\Delta f_j(\mathbf{x}^*)] &= \mathtt{E}\left[\left(\nabla f_j(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta\mathbf{x} \right)^2 \right] \\ &= \sigma^2 \sum_{i=1}^{n} \left(\left. \frac{\partial f_j(\mathbf{x})}{\partial x_i} \right|_{\mathbf{x}=\mathbf{x}^*} \right)^2. \end{aligned} \tag{5.3.10}$$

From (5.3.1) and (5.3.10) we express sensitivity of the j^{th} objective function to AWGN at $\mathbf{x}^*$ as follows.

$$\mathcal{S}_j(\mathbf{x}^*)|_{\text{AWGN}} := \sqrt{\sum_{i=1}^{n} \left(\left. \frac{\partial f_j(\mathbf{x})}{\partial x_i} \right|_{\mathbf{x}=\mathbf{x}^*} \right)^2}. \tag{5.3.11}$$

Thus for AWGN, we do not need to draw random samples in the neighbourhood of a solution to compute sensitivity subject to the following. First, the objective functions

are differentiable. Seconds, the perturbations in the variables are very small. Third, we use an approximation with Taylor series expansion considering only the first order terms.

Let us consider sensitivity to uniform noise. For this purpose, let us assume that $\Delta x_i \sim \mathcal{U}(-\delta, \delta); i = 1, 2, \cdots, n$, where δ is a very small positive value. In this case, $\mathtt{E}[\Delta x_i] = 0, i = 1, 2, \cdots, n$ and $\mathtt{var}[\Delta x_i] = \sigma^2 = \delta^2/3$ hold. Also, it is easy to see that (5.3.7); (5.3.8); and (5.3.9) hold. Therefore, (5.3.11) can be used for computing $\mathcal{S}_j(\cdot)$ with an approximation subject to the above three conditions. As the expressions of $\mathcal{S}(\cdot)$ with AWGN and with uniform noise are the same, in this Chapter, we do not experiment with uniform noise considering an approximation using the first-order Taylor series expansion.

5.3.3 Sensitivity to Multiplicative Noise

To model perturbations of variables with an MN, we assume that $\Delta x_i = x_i^* \nu(i), i = 1, 2, \cdots, n$, such that, $\nu(i) \sim \mathcal{N}(0, \sigma), i = 1, 2, \cdots, n$. Then, $\mathtt{E}[\Delta x_i] = 0, i = 1, 2, \cdots, n$, and $\mathtt{var}[\Delta f_j(\mathbf{x}^*)], j = 1, 2, \cdots, m$, can be expressed as

$$\begin{aligned} \mathtt{var}[\Delta f_j(\mathbf{x}^*)] &= \mathtt{E}\left[\left(\nabla f_j(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta \mathbf{x}\right)^2\right] \\ &= \sigma^2 \sum_{i=1}^{n} \left(\frac{\partial f_j(\mathbf{x})}{\partial x_i}\bigg|_{\mathbf{x}=\mathbf{x}^*} \right)^2 (x_i^*)^2 . \end{aligned} \tag{5.3.12}$$

From (5.3.1) and (5.3.12) we express sensitivity of the j^{th} objective function to MN at $\mathbf{x}^*$ as follows.

$$\mathcal{S}_j(\mathbf{x})|_{\text{MN}} \coloneqq \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f_j(\mathbf{x})}{\partial x_i}\bigg|_{\mathbf{x}=\mathbf{x}^*} \right)^2 (x_i^*)^2}. \tag{5.3.13}$$

In this fashion, similar to AWGN, for MN also we do not need to draw random samples in the neighbourhood of a given multiobjective solution provided that the Taylor series expansion using only the first-order term makes a good approximation.

5.3.4 A Generalized Definition of Sensitivity

When the noise added to the i^{th} variable has the standard deviation σ_i, i.e., $\mathtt{var}[\Delta x_i] = \sigma_i^2, i = 1, 2, \cdots, n$, we define generalized sensitivity of the j^{th} objective function $\left(\mathcal{S}_j^{\mathcal{G}}(\cdot)\right)$

at $\mathbf{x}^*$ as follows.

$$\mathcal{S}_j^{\mathcal{G}}(\mathbf{x}^*) := \frac{1}{n}\sum_{i=1}^{n}\frac{\sqrt{\texttt{var}\left[\Delta f_j(\mathbf{x}^*)\right]}}{\sigma_i}. \tag{5.3.14}$$

When $\sigma_i = \sigma, i = 1, 2, \cdots, n$, $\mathcal{S}_j^{\mathcal{G}}(\cdot)$ reduces to $\mathcal{S}_j(\cdot)$. Next, we define generalized sensitivity of an MOP at $\mathbf{x}^*$ as follows.

$$\mathcal{S}^{\mathcal{G}}(\mathbf{x}^*) := \phi(\mathcal{S}_1^{\mathcal{G}}(\mathbf{x}^*), \mathcal{S}_2^{\mathcal{G}}(\mathbf{x}^*), \cdots, \mathcal{S}_m^{\mathcal{G}}(\mathbf{x}^*)). \tag{5.3.15}$$

Here, $\phi(\cdot)$ is as defined earlier. Note that, there could be other ways to define a generalized version of sensitivity.

5.3.5 Using Sensitivity in Optimization

Next, we reformulate the MOP described in (1.1.1) using the following three approaches to incorporate sensitivity in the search process.

5.3.5.1 Approach I

In this approach, we incorporate sensitivity as an additional objective function in the multiobjective search process. As solving an MOP yields a set of nondominated solutions, this reformulated MOP provides solutions of the original MOP with different degrees of robustness. Approach I reformulates (1.1.1) as follows.

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathcal{V}}{\text{minimize}}\, \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x}), \mathcal{S}^{\mathcal{G}}(\mathbf{x}));\\
&\text{subject to}\\
&g_j(\mathbf{x}) \leq 0, j = 1, 2, \cdots, n_{\neq};\\
&h_k(\mathbf{x}) = 0, k = 1, 2, \cdots, n_{=};
\end{aligned} \tag{5.3.16}$$

5.3.5.2 Approach II

Here, we take into account sensitivity as an additional constraint in the multiobjective search process. Solving the reformulated MOP, we get a set of solutions of the original MOP with an upper limit on the degree of sensitivity. Approach II reformulates (1.1.1)

as follows.

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathcal{V}}{\text{minimize}}\, \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x}));\\
&\text{subject to}\\
&g_j(\mathbf{x}) \leq 0, j = 1, 2, \cdots, n_{\neq};\\
&h_k(\mathbf{x}) = 0, k = 1, 2, \cdots, n_{=};\\
&\mathcal{S}^{\mathcal{G}}(\mathbf{x}) < \eta^{\mathcal{S}}
\end{aligned} \tag{5.3.17}$$

5.3.5.3 Approach III

Finally in this approach, we add sensitivity of each objective function as an independent constraint to the multiobjective search process. Therefore, for an MOP with m objectives, m constraints are added in the reformulated MOP. Approach III reformulates (1.1.1) as follows.

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathcal{V}}{\text{minimize}}\, \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x}));\\
&\text{subject to}\\
&g_j(\mathbf{x}) \leq 0, j = 1, 2, \cdots, n_{\neq};\\
&h_k(\mathbf{x}) = 0, k = 1, 2, \cdots, n_{=};\\
&\mathcal{S}^{\mathcal{G}}{}_l(\mathbf{x}) < \eta_l^{\mathcal{S}}, l = 1, 2, \cdots, m.
\end{aligned} \tag{5.3.18}$$

5.4 Experiments and Results

5.4.1 Common Experimental Setting

5.4.1.1 Test Problems

In this Chapter, we use six TPs introduced in [63]. We denote these TPs as TP1, TP2, TP3, TP4, TP5, and TP6, respectively. The Appendix contains the details of these TPs and the partial derivatives of their objective functions. These derivatives are required for an approximation using Taylor series expansion of $\mathcal{S}(\cdot)$ when either AWGN or MN model of perturbation is considered. The first four TPs have two objectives and the last two TPs have three objectives. For all the TPs, throughout this work, we have considered the number of variables $(n) = 5$. Note that, some of the TPs are not differentiable at some boundary points, and hence, we have made minor changes in the TPs by reducing the range of the corresponding variables by a very small amount,

such that, this alteration preserves the nature of the TPs. A detailed description of these changes is provided in the Appendix.

5.4.1.2 Multiobjective Optimizer

We have used ASMiGA as a multiobjective optimizer throughout the following experiments. To obtain the solutions of an MOP, we solve it ten times using ASMiGA, and then, we choose the union of the obtained solutions. For each run, the operators and parameters used are as follows: number of function evaluations $= 50000$, maximum archive size $(N_{max}) = 200$, minimum archive size $(N_{min}) = 100$, selection ratio $(S_r) = 0.15$, differential evolution-3 crossover (with scaling factor $F = 0.5$ and crossover ratio $CR = 0.1$), and polynomial mutation (with $\eta_m = 50.0$).

5.4.1.3 Perturbation Strategy

We consider that a noise is either uniform, AWGN, or MN. When we employ an approximation using (5.3.11) or (5.3.13), respectively, considering AWGN or MN, we do not need to draw samples to compute $\mathcal{S}(\cdot)$. In the remaining part of this Chapter, when we mention that we have computed $\mathcal{S}(\cdot)$ with an "approximation", we mean that either (5.3.11) or (5.3.13) has been used depending upon the context. Below, we discuss the perturbation strategy that we employ when we do not use an approximation.

When an approximation is not used, irrespective of the type of noise, we generate 1000 n-dimensional $(n = 5)$ noise vectors. In every case, we use a parameter δ to denote the degree of perturbations. When uniform noises are considered, noise vectors are drawn using LH sampling inside the n-dimensional hypercube with vertices $(\pm\delta, \pm\delta, \cdots, \pm\delta)^T \in \mathbb{R}^n$. Note that, the uniform distribution has zero mean and a standard deviation of $2\delta/\sqrt{12}$. Similarly, when AWGN is considered, each dimension of n-dimensional noise vectors has zero mean and standard deviation $\sigma = 2\delta/\sqrt{12}$. In the same fashion, when MN is considered, to perturb a solution $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T \in \mathcal{V} \subseteq \mathbb{R}^n$, we generate 1000 perturbation vectors $\Delta\mathbf{x}_k = (x_1\nu_{k1}(\cdot), x_2\nu_{k2}(\cdot), \cdots, x_n\nu_{kn}(\cdot))^T \in \mathbb{R}^n$, $k = 1, 2, \cdots, 1000$, such that, $\nu_{ki}(\cdot)$, $i = 1, 2, \cdots, n$ is a white Gaussian noise with zero mean and standard deviation $\sigma = 2\delta/\sqrt{12}$. In our experiments, irrespective of the type of noise, we choose δ in such a manner that the standard deviations (σs) of the perturbations remain the same. If after applying a noise vector the perturbed solution goes beyond the lower limit or the upper limit of any variable, that particular noise vector is ignored. Finally $\mathcal{S}(\cdot)$ is calculated using the

remaining perturbed points. When, $\mathcal{S}(\cdot)$ is computed, we assume that the standard deviation of the perturbations is known. We also note here that, all primary choices of δ in this Chapter are the same as in [63]. However, unlike [63], for simplicity here we consider $\sigma_i = \sigma,\ i = 1, 2, \cdots, n$, i.e., the same variance of perturbations in all the variables. When investigating on Approach I, for different TPs, the choices of δ are as follows: δ_1, δ_2, δ_3, and δ_4 for TP1; δ_5, δ_6, δ_7, and δ_1 for TP2; δ_8 for TP3, TP4, and TP6; and δ_4 for TP5; where $\delta_1 = 0.007$, $\delta_2 = 0.008$, $\delta_3 = 0.009$, $\delta_4 = 0.010$, $\delta_5 = 0.004$, $\delta_6 = 0.005$, $\delta_7 = 0.006$, and $\delta_8 = 0.030$. Note that, corresponding each δ_i, there is a distinct $\sigma_i = 2\delta_i/\sqrt{12}$ that is used for AWGN and MN. When investigating on Approach II, we use only TP1 with δ_4. Moreover, to investigate deeper, we also use some other values of δ for some cases. Throughout this work, we use LH sampling to generate uniform noises, because it is a frequently used strategy for generating uniform noises and in [63], Deb et al. also used LH sampling for this purpose.

5.4.2 Sensitivity: a Measure of Deviation and Robustness

At first, we want to examine if $\mathcal{S}(\cdot)$ works as a measure of deviation. To inspect this, we generate three sets of noises, each containing 1000 n-dimensional $(n = 5)$ perturbation vectors, considering uniform noise $(\delta = 0.010)$, AWGN $(\sigma = 5.7735 \times 10^{-03})$, and MN $(\sigma = 5.7735 \times 10^{-03})$. Note that, σ is chosen as $2 \times \delta/\sqrt{12}$. Now, we consider the type I robust solutions obtained in our previous experiment discussed in Section 5.2.2. For each of these solutions, using each of these three sets of perturbation vectors, we compute the corresponding $\mathcal{S}(\cdot)$s and $\texttt{MoD}[\cdot,\cdot]$s. In Fig. 5.4 we plot $\mathcal{S}(\cdot)$ and $\texttt{MoD}[\cdot]$ for uniform distribution. From Fig. 5.4, we observe that $\mathcal{S}(\cdot)$ changes almost linearly with $\texttt{MoD}[\cdot,\cdot]$. This is also true for AWGN and MN (see Fig. A-1; a figure number with a suffix A- indicates that the figure is included in the Appendix). $\mathcal{S}(\cdot)$, consequently, can be used as a measure of deviation. Next, we show the solutions with uniform noise in a three-dimensional stem plot in Fig. 5.5. The third dimension of Fig. 5.5 is the $\mathcal{S}(\cdot)$s of the solutions considering uniform noise with $\delta = 0.010$. Figure 5.5 shows that with a decrease in $f_1(\cdot)$ and an increase in $f_2(\cdot)$, $\mathcal{S}(\cdot)$ increases for this set of solutions. This result is consistent with the results provided in [63], because in [63] Deb et. al has observed that on TP1 with a decrease in $f_1(\cdot)$ and an increase in $f_2(\cdot)$ the robust PF goes further away from the original PF (See Fig. 8 and Fig. 9 of [63]). Note that, this observation can also be made from Fig. 5.1. From this experiment we validate that, given a set of solutions (may be a set of robust solutions of type I), $\mathcal{S}(\cdot)$ can be used to quantify their robustness and to distinguish the solutions in terms of robustness.

Now, we answer an important question. Instead of $\mathcal{S}(\cdot)$, why cannot we use

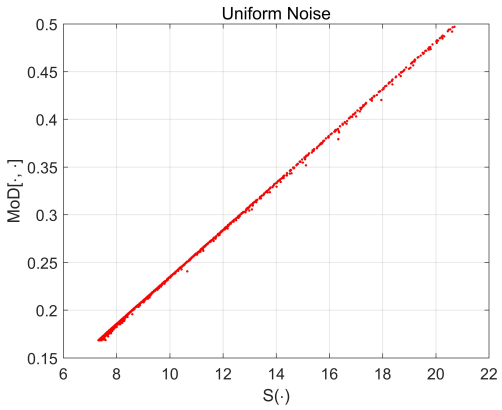


Figure 5.4: $\texttt{MoD}[\cdot,\cdot]$-versus-$\mathcal{S}(\cdot)$ plots of the obtained type I robust solutions for uniform noise with $\delta = 0.010$.

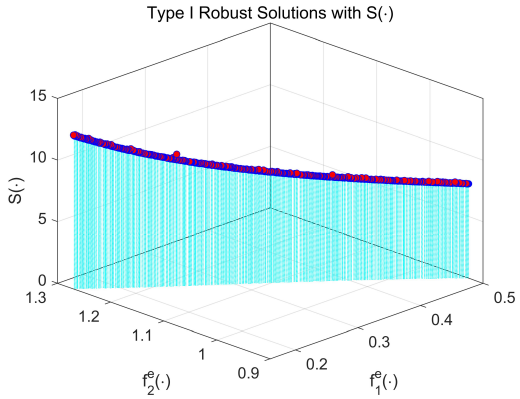


Figure 5.5: Three-dimensional stem plot of the obtained type I robust solutions for uniform noise with $\delta = 0.010$.

$\texttt{MoD}[\cdot,\cdot]$ itself as a measure? There are two reasons. First, if we use $\mathcal{S}(\cdot)$ with an approximation, it would cause a huge reduction in the computational cost of explicit robustness handling, especially when each function evaluation is costly. However, if we use $\texttt{MoD}[\cdot,\cdot]$, we cannot do that. Second, $\texttt{MoD}[\cdot,\cdot]$ does not take into account any statistical characteristic of the noise, whereas, $\mathcal{S}(\cdot)$ considers that. $\mathcal{S}(\cdot)$, consequently, is a better measure in terms of our intuition. However, we agree that $\texttt{MoD}[\cdot,\cdot]$ can also be used as a measure of robustness/sensitivity.

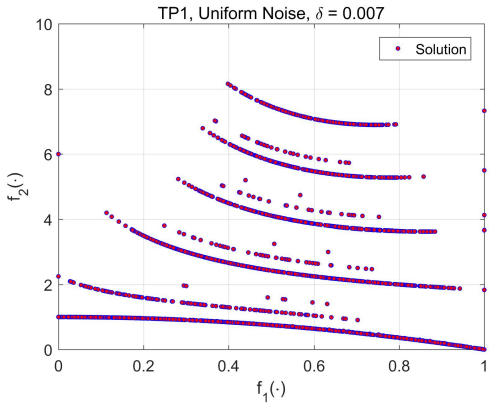


Figure 5.6: Scatter plots of the obtained solutions on TP1 using Approach I for uniform noise with δ_1.

5.4.3 Investigations on Approach I

To investigate the behaviour of Approach I, we reformulate six TPs described in Section 5.4.1.1 using the optimization strategy discussed in Section 5.4.1.2 with the perturbation strategies provided in Section 5.4.1.3. Then, we examine the obtained solutions as discussed next.

5.4.3.1 Experiments on Test Problem 1 (TP1)

At first, we consider Approach I on TP1 with uniform noise. Figures 5.6 and 5.7, respectively, present a two dimensional scatter plot (in short 2D) and a three-dimensional stem plot (in short 3D) of the obtained solutions on TP1 using Approach I with perturbation δ_1. In each of Fig. A-2 (2D) and Fig. A-3 (3D), we illustrate the solutions obtained with δ_2, δ_3, and δ_4. These 2D plots illustrate that, irrespective of the choices of δ, the proposed method could find the same local fronts along with the original PF (to see the original PF of TP1, see Fig. 5.1). With an increase in δ, however, the local fronts grow, especially in the top-left region of the 2D plots. From the 3D plots, we observe that solutions away from the original PF have lower values of $\mathcal{S}(\cdot)$. Moreover, for a local front, the $\mathcal{S}(\cdot)$s of its solutions increase with a decrease in $f_1(\cdot)$. With an increase in δ, the values of $\mathcal{S}(\cdot)$ increase for the solutions of the same local front. Both of these observations are intuitive considering similar arguments provided in Section 5.4.2.

When we consider AWGN on TP1, the obtained solutions for different values of δs (σs) have been shown in Fig. A-4 using 2D plots and in Fig. A-5 using 3D plots.

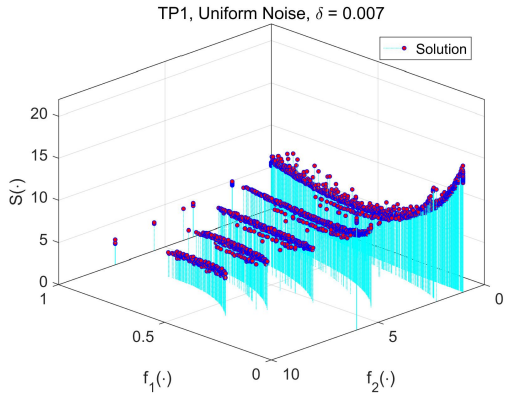


Figure 5.7: Three-dimensional stem plot of the obtained solutions on TP1 using Approach I for uniform noise with δ_1 .

The results obtained with AWGN is quite similar to the results obtained with uniform noise. However, for each value of δ, compared to Fig. A-2, we observe that, in Fig. A-4 more solutions are obtained in the top-left regions of the corresponding subfigures. From these two figures, we also observe that for the same value of δ, the lengths of the local fronts are larger with AWGN compared to the case with uniform noise. As mentioned earlier, in both the cases we use the same values of standard deviations. This limited experiment, therefore, indicates that for a given standard deviation, a search considering AWGN is likely to find larger local fronts compared to a search considering uniform noise. Comparing Fig. A-3 with Fig. A-5, we observe that, for each value of δ, for the solutions of the same local front usually $\mathcal{S}(\cdot)$ is higher with AWGN compared to the case with uniform noise. When we compute $\mathcal{S}(\cdot)$ with an approximation considering AWGN, the obtained solutions are shown in Fig. A-6. Here, we make following two important observations:

1. The search mechanism could find solutions throughout the original PF, but could not find any solution form any local front.

2. In Fig. A-6b, for the solutions corresponding to the original PF, with an increase in $f_1(\cdot)$ and a decrease in $f_2(\cdot)$, the values of $\mathcal{S}(\cdot)$ increase. On the contrary, in the subfigures of Fig. A-5, with an increase in $f_1(\cdot)$ and a decrease in $f_2(\cdot)$, the values of $\mathcal{S}(\cdot)$ decrease for the solutions corresponding to the original PF.

A probable reason behind the first observation could be as follows. From Fig. A-5 we found that with a decrease in δ, the lengths of the local fronts reduce, and hence, when the perturbation is very small, the search process may find no solution from any

of the local fronts. This incident, however, may be problem dependent. The second observation, at the first glance, is sufficiently counterintuitive. To investigate deeper about these observations, at first, we find solutions on TP1 considering AWGN with $\delta = 1.0 \times 10^{-06}$. Note that, the other choices of δ are inspired by the work in [63] that are much larger than this choice. Therefore, in this case, we expect the obtained solutions to be like the solutions in Fig. A-6. This is because our approximation considers a very small change in the variable and in our opinion $\delta = 1.0 \times 10^{-06}$ is small enough to realize that. We illustrate the obtained solutions in Fig. A-7. The nature of the solutions in Fig. A-7 is quite similar to the solutions in Fig. A-6. Therefore, our reasoning regarding the first observation seems true. The second observation still remains somewhat counterintuitive. Therefore, to look into this issue, we perform the following experiment. For TP1, the set $\{\mathbf{x}|x_1 \in [0,1]; x_i = 0, i = 2,3,\cdots,n\}$ constitutes the PS. We consider 1000 uniformly distributed solutions along x_1, such that, the set covers the entire PS. Now, we compute $\mathcal{S}(\cdot)$s for all of these solutions considering AWGN with (i) Taylor approximation, (ii) $\delta = 1.0 \times 10^{-05}$, (iii) $\delta = 1.0 \times 10^{-04}$, (iv) $\delta = 2.0 \times 10^{-04}$, (v) $\delta = 5.0 \times 10^{-04}$, and (vi) $\delta = 1.0 \times 10^{-03}$. In Figs. A-8a to A-8f, we plot the objective values of these solutions along with $\mathcal{S}(\cdot)$ for these six cases, respectively. From these figures, we observe that with an increase in δ, the nature of $\mathcal{S}(\cdot)$ of the solutions in the PS changes gradually, and the direction of change in $\mathcal{S}(\cdot)$ when traversing along the same front reverses, at least for a part of the front. In Fig. A-9, we plot $\texttt{MoD}[\cdot,\cdot]$-versus-$\mathcal{S}(\cdot)$ of the obtained solutions for AWGN with $\sigma = 0.010$. This plot confirms that $\texttt{MoD}[\cdot,\cdot]$ changes linearly with $\mathcal{S}(\cdot)$ (they have a Pearson's correlation coefficient of 0.9995). Thus, when AWGN is considered, for a problem δ needs to be chosen based on the characteristics of the problems.

Now, We consider MN. Figures A-10 and A-11 show the 2D and 3D plots of the obtained solutions for the four different choices of δ, respectively. In all of these subfigures, we observe a triangular cluster of solutions near $f_1(\cdot) = 1$. Figure A-11 reveals that the solutions in this cluster have a varying degree of $\mathcal{S}(\cdot)$. The remaining solutions visually recline on the original PF, and for them, with an increase in $f_1(\cdot)$ and a decrease in $f_2(\cdot)$, $\mathcal{S}(\cdot)$ increases. Here, we notice that visually the nature of the obtained solutions with different values of δ is quite similar. In Figs. A-12a and A-12b, respectively, we show the 2D and 3D plots of the obtained solutions with the Taylor approximation for MN. In this case also solutions are found throughout the original PF. However, in this case, the search did not find any cluster of solutions near $f_1(\cdot) = 1$. Is this because the chosen δs are high? To check this we experimented with $\delta = 1.0 \times 10^{-06}$ and found that it is indeed the case.

5.4.3.2 Experiments on Test Problem 2 (TP2)

To examine Approach I on TP2 we use eight figures. Figures A-13 and A-14 correspond to uniform noise, Figs. A-15 and A-16 correspond to AWGN, Fig. A-17 corresponds to AWGN with the Taylor approximation of $\mathcal{S}(\cdot)$, Figs. A-18 and A-19 correspond to MN, and Fig. A-20 corresponds to MN with the Taylor approximation of $\mathcal{S}(\cdot)$. Each of Figs. A-13 to A-16, A-18 and A-19 has four subfigures, corresponding to δ_5, δ_6, δ_7, and δ_1, respectively.

From Figs. A-13, A-14, A-18 and A-19, we make the following observations when uniform noise and AWGN are considered. First, irrespective of the choice of δ, the evolutionary search could find solutions corresponding to the same set of local fronts along with the entire global front. Second, with an increase in δ, the lengths of the local fronts increase. Third, with an increase in δ, $\mathcal{S}(\cdot)$s of the solutions belonging to the same front increase. Fourth, with an increase in the distances between a local front and the original front, the $\mathcal{S}(\cdot)$s of its solutions decrease. Figure A-17 illustrates that when we compute $\mathcal{S}(\cdot)$ with an approximation considering AWGN, solutions are obtained throughout the original PF and no solution is found from any local front. The variations of $\mathcal{S}(\cdot)$ with variations of $f_1(\cdot)$ and $f_2(\cdot)$ are similar to that of TP1. This is intuitive because the natures of these two TPs are somewhat similar.

5.4.3.3 Experiments on Test Problem 3 (TP3)

Here, we consider five cases: (i) uniform noise with δ_8, (ii) AWGN with δ_8, (iii) Taylor approximation for AWGN, (iv) MN with δ_8, and (v) Taylor approximation for MN. The obtained solutions for these five cases are illustrated in Figs. A-21 to A-25, respectively. From these figures, we observe that, for every case except the case when an approximation with MN is used, the search process could find solutions from a local front and from the global front. From the corresponding 3D plots of these figures, we observe that the solutions belonging to the local front have a smaller degree of $\mathcal{S}(\cdot)$ compared to that of the global front.

5.4.3.4 Experiments on Test Problem 4 (TP4)

Here also we consider the same five cases examined on TP3. We illustrate the obtained solutions on TP4 in Figs. A-26 to A-30, respectively. The organization of these figures are the same as of the figures that correspond to TP3. From these five figures, we observe that, for each of the cases, the evolutionary search scheme found solutions throughout the original PF. Additionally, a cluster of solutions is found near $f_1(\cdot) = 1$.

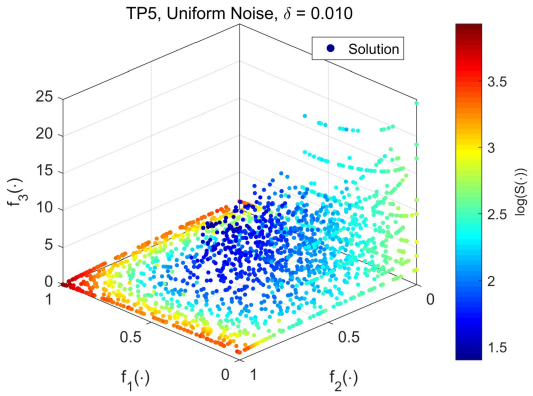


Figure 5.8: Four-dimensional scatter plots of the obtained solutions on TP5 using Approach I for uniform noise with δ_4.

5.4.3.5 Experiments on Test Problem 5 (TP5)

For TP5, we consider five cases: (i) uniform noise with δ_4, (ii) AWGN with δ_4, (iii) Taylor approximation for AWGN, (iv) MN with δ_4, and (v) Taylor approximation for MN. The obtained solutions for the first case has been illustrated in Fig. 5.8 and the obtained solutions for the other four cases have been illustrated in Fig. A-31. Figure 5.8 and each of the subfigures of Fig. A-31 are four-dimensional scatter plots. Here, the fourth dimension is $\mathcal{S}(\cdot)$ in the corresponding scenario, which has been represented using colours. From Fig. A-31, we observe that when uniform noise or AWGN is considered, the search process could find solutions from several local fronts along with the entire original PF. When (i) Taylor approximation is used for AWGN, (ii) MN with δ_4 is used, and (ii) Taylor approximation is used for MN, $\mathcal{S}(\cdot)$ decreases with an increase in $f_3(\cdot)$. When an approximation considering AWGN is used, a very small number of solutions are found from any local front and all the obtained solutions correspond to the original PF. When MN is considered both with and without approximation, solutions are found from a local PF as well as from the original PF.

5.4.3.6 Experiments on Test Problem 6 (TP6)

For TP6, we consider the same five cases that we have used for TP3. The obtained solutions for these five cases are depicted in Fig. A-32. From Fig. A-32, we observe that in every case the search scheme could find solutions from the original PF as well as from several local fronts. When AWGN, MN, and Taylor approximation considering MN are used, there are notable differences in terms of $\mathcal{S}(\cdot)$ for the solutions that cor-

respond to different fronts. With AWGN, the solutions corresponding to the original PF have a higher degree of $\mathcal{S}(\cdot)$ compared to the solutions of the other three fronts. When Taylor approximation considering AWGN is used, solutions associated with all three fronts have a similar degree of $\mathcal{S}(\cdot)$.

5.4.4 Investigations on Approach II

To examine Approach II on TP1, we consider five cases: (i) uniform noise with δ_4, (ii) AWGN with δ_4, (iii) Taylor approximation for AWGN, (iv) MN with δ_4, and (v) Taylor approximation for MN. For each of these five cases, we select the mean values of the $\mathcal{S}(\cdot)$s of the obtained solutions using Approach I in the corresponding scenario. The obtained mean values are 8.3375, 13.7470, 1.3135, 0.9117, and 0.7175, respectively. We use these values as the limiting parameter $\eta^{\mathcal{S}}$ in (5.3.17) and in a similar way as in Approach I we find solutions. Note that, for a real life problem usually our tolerance to sensitivity $(\eta^{\mathcal{S}})$ will be dictated by the problem.

In Fig. A-33, we depict the obtained solutions for uniform noise with δ_4. From Fig. A-33, we observe that when uniform noise is considered, primarily there are solutions from three fronts, including one corresponding to the (partial) original PF. However, the search process rejected solutions with higher values of $\mathcal{S}(\cdot)$ than the given value of $\eta^{\mathcal{S}}$ in the left parts of these fronts. This is what we wanted. Next we show the obtained solutions for AWGN with δ_4 in Fig. 5.9 (see Fig. A-34 for the corresponding 3D plot). Figure 5.9 shows that the evolutionary search process found solutions from the right side of the original front approximately in the region $f_1(\cdot) > 0.2$. However, when Taylor approximation for AWGN is used, the search process found three local fronts (see Fig. A-35). Figure A-35b depicts that when traversed along these local fronts, the rates of change (in a loose sense the gradient) of $\mathcal{S}(\cdot)$ are notably different.

Now, we illustrate the solutions for MN with δ_4 in Fig. 5.10 using a 2D plot (see Fig. A-36 for the corresponding 3D plot). Figures 5.10 and A-36 reveal the following: (i) solutions corresponding to a part of the original PF are found, (ii) the solutions from the right side of the front are removed due to high sensitivity, and (iii) the values of $\mathcal{S}(\cdot)$ increase with an increase in $f_1(\cdot)$. The solutions obtained with the Taylor approximation for MN shows a similar behavior (see Fig. A-37b). It is noteworthy that when uniform noise with $\delta = 0.010$ and AWGN with $\delta = 0.010$ are considered, the left part of the original PF is removed. This observation is consistent with our observation made for Approach I. Therefore, when traversing along a local front, $\mathcal{S}(\cdot)$ corresponding to uniform noise and AWGN may increase but the $\mathcal{S}(\cdot)$ corresponding to MN may decrease. A plausible explanation of this observation is as follows. For the PF of TP1 $x_1 \in [0, 1]$

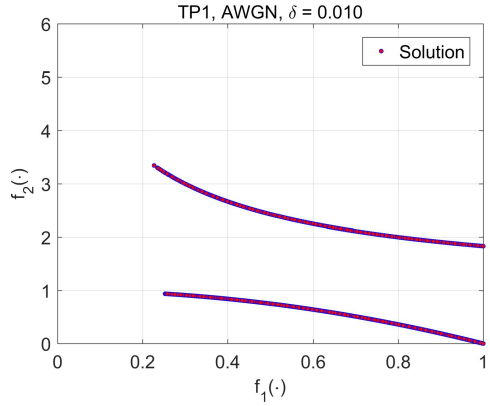


Figure 5.9: Scatter plot of the obtained solutions on TP1 using Approach II for AWGN with δ_4.

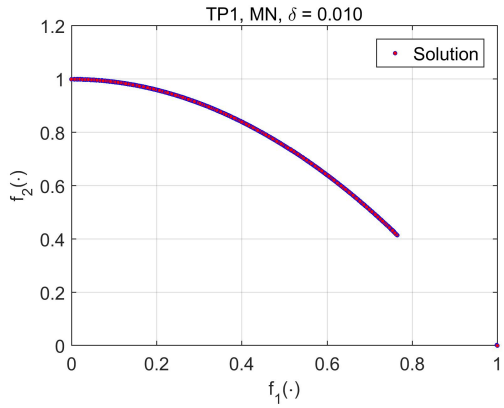


Figure 5.10: Scatter plots of the obtained solutions on TP1 using Approach II for MN with δ_4.

and $x_i = 0,\ i = 2, 3, \cdots, n$. Even with a large value of σ, for $x_1 \in [0, 1]$, the perturbations in the variable space for MN are small. This can be viewed as somewhat similar to the perturbations for AWGN with a small value of σ, because a multiplication by $x_1 \in [0, 1]$ reduces the magnitude of the perturbations. The minimum, the maximum, the mean, and the standard deviation of the $\texttt{MoD}[\cdot, \cdot]$s of the solutions obtained for MN with $\sigma = 0.010$ are 9.6777×10^{-09}, 1.0388×10^{-02}, 5.7982×10^{-03}, and 2.7859×10^{-03}, respectively. The same four statistics of the solutions obtained for AWGN with the same σ are 1.7139×10^{-01}, 4.8568×10^{-01}, 2.6215×10^{-01}, and 9.4096×10^{-02}, respectively. As a smaller value of $\texttt{MoD}[\cdot, \cdot]$ is caused by a smaller degree of perturbations, these statistics validate our argument.

5.5 Conclusions and Discussions

In this Chapter, we have proposed a new measure of sensitivity ($\mathcal{S}(\cdot)$) in the context of MOPs. With an increase in this measure, the sensitivity of the multiobjective solutions increases and their robustness decreases. We have provided two ways to approximate $\mathcal{S}(\cdot)$ subject to three conditions. We have also proposed three approaches to reformulate MOPs to take into account $\mathcal{S}(\cdot)$ in the evolutionary search process. The first reformulation provides solutions with varying sensitivity/robustness. The other two reformulated MOPs produce solutions with $\mathcal{S}(\cdot)$ less than a predefined threshold.

We have extensively investigated the behaviour of the first approach on six TPs and have made several observations. Some of them are as follows. First, for TP1 and TP2, with an increase in the standard deviation of the input perturbations, more solutions are found on the local fronts. Second, for TP1 and TP2, when the standard deviation of the input noise increases, the values of $\mathcal{S}(\cdot)$ for the solutions from the same local fronts increase. Third, when the standard deviation is very small, as expected the obtained solutions are similar to the solutions obtained with an approximation. Fourth, for a given standard deviation, usually, the values of $\mathcal{S}(\cdot)$ is higher when AWGN is considered compared to that of considering uniform noise. We have also inspected the behaviour of the second approach on TP1. We found that for TP1 while traversing along a front the values of $\mathcal{S}(\cdot)$ increase for both uniform noise and AWGN, but for MN under the same condition $\mathcal{S}(\cdot)$ decrease. Therefore, for TP1, different sides of the fronts are removed by the constraint for different types of noises.

Unless the problem characterizes the nature of the noise, the type of noise should be chosen judiciously. This is because, the solutions obtained considering uniform noise or AWGN maybe notably different from the solutions obtained considering MN. Moreover, for a chosen type of noise, the value of the parameter that characterizes the perturbations (δ or σ) should be chosen wisely. This is because, for a chosen type of noise, two sets of solutions obtained for two different choices of δ (or σ) may differ significantly. Furthermore, an approximation using the first-order Taylor series expansion should be used only when the weight perturbations are small.

We could not compare the proposed measure with any existing measure primarily because of two reasons. First, to the best of our knowledge, there is no work in the literature that defines/uses a similar measure in the context of MOPs. The closest work to this work probably is in [173], which has defined a measure for the same purpose in the context of single objective optimization, whereas, our work focuses on MOPs. Second, there is a lack of straightforward way for comparison.

Chapter 6

Conclusions and Future Scopes

6.1 Conclusions

In this thesis, we address four research problems related to decision making using multiobjective evolutionary approaches. In Chapter 2, we propose a method for simultaneous feature selection (FS) and classification. In Chapter 3, we propose a method of feature extraction and selection for designing parsimonious classifiers; while in Chapter 4 work, in the context of fuzzy group decision making for a multiobjective optimization problem (FGDM-MOP), we propose a new measure, called robust consensus, and incorporate it in the search process. In Chapter 5, a new measure of sensitivity is proposed that can assess the sensitivity, and hence, robustness of a multiobjective solution. We propose three approaches which take into account this measure of sensitivity in the search process.

In Chapter 2, we have proposed a multiobjective genetic programming (MOGP), called archive based steady state micro genetic programming (ASMiGP), to evolve diverse sets of binary classifiers to solve multi-class classification problems. We have also performed simultaneous FS and rule extraction during the genetic evolution. Here, we have created c sets of diverse ensembles, where each ensemble represents diverse classifiers for one class. In the proposed strategy, we have evolved c distinct species in parallel, which try to learn distinct patterns and we have not allowed any inter-species gene exchange. This attribute makes each species different from the other species. We have compared our method with four classification methods in conjunction with seven FS methods including use of the all-feature set on nineteen (eight microarray and eleven text) data sets. We have found that for 80.17% cases the proposed method outperformed others, and the improvement in performance was statistically

significant compared to all but one comparing methods. The overall performance of the proposed method has been found to be better on text data sets compared to that on microarray data sets. Our experiments have shown that when there are enough points in each class so that each species can successfully learn its target pattern, the proposed method works better. For text data sets, our method performs noticeably better than the other methods, particularly when number of classes is high (ten or more than that).

In the next Chapter, we have investigated the capability of genetic programming (GP) to extract useful features that can make the data linearly separable. We have called such features as linearly separable features. Here, we have also proposed a MOGP called archive based micro genetic programming-2 (ASMiGP-2). Based on the ASMiGP-2, we have developed an embedded FS and feature extraction (FE) strategy for generating an ensemble of parsimonious classifiers that use only linearly separable features. For a c class problem, the classifier is an ensemble of c archives, where each archive is an ensemble of binary classifiers. Each binary classifier is composed of a binary tree (genetic program) and a support vector machine (SVM) with a linear kernel. In each binary classifier, the genetic program selects and extracts a set of features and the corresponding SVM operates in the modified feature space. In this regard, we have proposed useful strategies for FS, archive initialization, crossover, and mutation. We have tested the proposed method on three artificial data sets to examine its capability to extract linearly separable features. Experimental results have shown that our method can successfully extract linearly separable features both with and without data normalization. We have compared our method with 34 algorithms on 18 data sets and have found that our method outperforms the comparing method for 75.79% comparing cases. Using appropriate statistical tests, we have found that the performance of our method is significantly better than 58.82% of the comparing methods. We have observed that there is no significant difference in the performance between the proposed method and the remaining 41.18% comparing methods. Lastly, we have found that our method can find rules with higher parsimony compared to the two GP-based methods discussed in Chapter 2.

In a multiobjective decision making problem when different experts provide their preferences, finding of optimal solutions satisfying all experts becomes challenging. We have deal with this problem in Chapter 4. We have proposed a framework to define *consensus* to measure the level of mutual agreement among a set of decision makers (DMs) for a given fuzzy group decision making for multiobjective optimization problem (FGDM-MOP). This framework can be used to define consensus in many ways. We have also defined an indicator, called *robust consensus*, to measure the ro-

bustness of a solution to its degree of consensus. In this context, we have proposed two different ways (Approach I and Approach II) to reformulate a given multiobjective optimization problem (MOP) so that solutions of the reformulated problem are also robust to their degree of consensus. We have studied the behaviour of these definitions and reformulations when the preferences are provided in the objective space. We have also investigated the changes in the nature of solutions when the specificities of the preferences provided by the DMs change. The effect of the changes in the weights or the importance of the DMs is also studied. From our limited investigation we have found that, the choice of membership functions has a significant impact on the degree of consensus and can make the outcomes differ significantly. If one wants to get a diverse set of solutions compromising the degree of consensus, Gaussian membership functions would be preferred over triangular membership functions. On the other hand, if we prefer to get a small set of solutions with a higher degree of consensus, then triangular membership functions may be preferred. We also have observed that the effect of specificity on robust consensus and the effect of weights on consensus are in accordance with our intuitions. We note here that one may prefer Approach II over Approach I if she wants the weights to play a stronger role. Further, one can control the nature of the influence of the weights to some extent by choosing a suitable weighted aggregation operator $(\psi(\cdot))$.

In Chapter 5, we have proposed a new measure of sensitivity in the context of MOPs. With an increase in this measure, the sensitivity of multiobjective solutions increases and their robustness decreases. We have provided three ways to approximate the proposed sensitivity measure considering either uniform noise, additive white Gaussian noise (AWGN), or multiplicative noise (MN) using Taylor series expansion. We have given three different reformulations of the MOPs taking into account this measure in the evolutionary search process. In the first approach, the reformulated MOP provides solutions with varying sensitivity/robustness. For the other two approaches, the reformulated MOP produces solutions that have a maximum degree of sensitivity, and hence, a minimum degree of robustness. We have investigated the behaviour of the first approach on six test problems (TPs) and have made several observations. Some of them are as follows. First, for test problem 1 (TP1) and test problem (TP2), more solutions appear on the local front with an increase in the standard deviation of the input perturbations. Second, for the same two TPs, the values of $\mathcal{S}(\cdot)$ for the solutions from the same local fronts increase with an increase in the standard deviation of the input noise. Third, the obtained solutions are similar to the solutions obtained with an approximation when the standard deviation is very small. Fourth,

usually the values of $\mathcal{S}(\cdot)$ is higher when AWGN is considered compared to that of considering uniform noise for a given standard deviation. We have also inspected the behaviour of the second approach on TP1. We found that, for this test problem, while traversing along a front the values of $\mathcal{S}(\cdot)$ increase for both uniform noise and AWGN, but for MN under the same condition $\mathcal{S}(\cdot)$ decrease. Therefore, for this TP, different types of noises remove different sides of the fronts.

6.2 Limitations and Future Scopes

Similar to most of the works, the works presented in this thesis have some limitations. Below, we discuss them along with possible future scopes to address them.

In all the data sets used for the experiments in Chapter 2, the number of samples were not very big. The proposed method may require a significant amount of time on a data set with a really large number samples. When the number of classes is high, and we have a limited parallel processing capability, the method may also take a substantial amount of time. With today's high performance computing technologies, however, these may not be considered really crucial shortcomings. However, to address this issue, the method can be modified to a stepwise learning, so that, it can be applied to big data. In a stepwise learning, the entire evolution can be divided into a number of steps. In the first step of the evolution, we can use a small subset of the training data. As the evolution moves to higher and higher steps, we can use more and more data and in the last step we can use the entire data. Our method can be extended to multi-label classification problems, where each sample may belong to more than one class. This may require defining new fitness and unfitness values.

The work presented in Chapter 3 requires training an SVM for each function evaluation and that makes the process computationally expensive, especially when the number of samples is large. However, this may possibly be addressed by softwares like TensorFlow. The proposed method requires several parameters to be chosen, which we have done based on a limited set of ad-hoc experiments. The use of a cross-validation type framework would be useful here. But, we could not investigate these issues due to a high computational requirement. Instead of using three objectives, it might be possible to have a method with a single objective. These issues need further investigations.

In our work on robust consensus, we have not investigated the scalability of the proposed methods in terms of the number of DMs and the number of objectives / variables. Although, we can assess an individual solution, it would be useful to define an

indicator to measure the quality of the obtained *set of solutions*. This is important, because, when we deal with a higher dimensional objective/variable space, it becomes difficult to assess the solutions visually. In future we like to make a detailed investigation on the performance of the variance based formulation as well as when the preferences are in the variable space. We also plan to use the FGDM-MOP considered in Chapter 4 in some real-world problems. For example, in a reactive dispatch problem operators (DMs) want to minimize two objectives: power loss and voltage deviation. For this problem, different operators may (usually will) have different preferences. Such a problem can be dealt with FGDM-MOP.

In Chapter 5, we have primarily experimented with Approach I and tested Approach II on one test problem. We intend to make an exhaustive investigation on Approach II and Approach III. Although the formulation is quite general, here we have considered the same standard deviation of noise for each variable. Further investigations on these issues may lead to interesting findings. In future, we plan to use our sensitivity based robust multiobjective optimization method for the reactive power dispatch problem.

Appendix

A Test Problems

A.1 Test Problem 1 (TP1)

This is the first bi-objective test problem introduced in [63].

$$
\begin{aligned}
&\text{minimize } f_1(\mathbf{x}) = x_1;\\
&\text{minimize } f_2(\mathbf{x}) = H\left(x_1\right) + G\left(\mathbf{x}\right) \cdot S\left(x_1\right);\\
&\text{subject to } 0 \leq x_1 \leq 1;\ -1 \leq x_j \leq 1,\ j = 2, 3, \cdots, n;\\
&\quad \text{where}\\
&\qquad H\left(x_1\right) = 1 - x_1^2;\\
&\qquad G\left(\mathbf{x}\right) = \sum_{j=2}^{n} \left(10 + x_j^2 - 10\cos\left(4\pi x_j\right)\right);\\
&\qquad S\left(x_1\right) = \frac{1}{0.2 + x_1} + x_1^2. \qquad \text{(A-1)}
\end{aligned}
$$

Both the objective functions of this test problem are differentiable. For this test problem, following holds true.

$$
\begin{aligned}
&\frac{\partial f_1(\mathbf{x})}{\partial x_1} = 1;\\
&\frac{\partial f_1(\mathbf{x})}{\partial x_j} = 0, j = 2, 3, \cdots, n;\\
&\frac{\partial f_2(\mathbf{x})}{\partial x_j} = \frac{\partial H(x_1)}{\partial x_j} + \frac{\partial G(\mathbf{x})}{\partial x_j} \cdot S(x_1) + G(\mathbf{x}) \cdot \frac{\partial S(x_1)}{\partial x_j}, j = 1, 2, \cdots, n;
\end{aligned}
$$

$$\frac{\partial H(x_1)}{\partial x_1} = -2x_1;$$
$$\frac{\partial H(x_1)}{\partial x_j} = 0, j = 2, 3, \cdots, n;$$
$$\frac{\partial G(\mathbf{x})}{\partial x_1} = 0;$$
$$\frac{\partial G(\mathbf{x})}{\partial x_j} = 2x_j + 40\pi \sin(4\pi x_j), j = 2, 3, \cdots, n;$$
$$\frac{\partial S(x_1)}{\partial x_1} = -\frac{1}{(0.2 + x_1)^2} + 2x_1;$$
$$\frac{\partial S(x_1)}{\partial x_j} = 0, j = 2, 3, \cdots, n. \tag{A-2}$$

A.2 Test Problem 2 (TP2)

This is the second bi-objective test problem introduced in [63].

$$\begin{aligned}
&\text{minimize } f_1(\mathbf{x}) = x_1;\\
&\text{minimize } f_2(\mathbf{x}) = H(x_1) + G(\mathbf{x}) \cdot S(x_1);\\
&\text{subject to } 0 \leq x_1 \leq 1;\ -1 \leq x_j \leq 1,\ j = 2, 3, \cdots, n;\\
&\quad \text{where}\\
&\qquad H(x_1) = 1 - x_1^2;\\
&\qquad G(\mathbf{x}) = \sum_{j=2}^{n} \left(10 + x_j^2 - 10\cos\left(4\pi x_j\right)\right);\\
&\qquad S(x_1) = \frac{1}{0.2 + x_1} + 10x_1^2.
\end{aligned} \tag{A-3}$$

Both the objective functions of this test problem are differentiable. For this test problem, following holds true.

$$\frac{\partial f_1(\mathbf{x})}{\partial x_1} = 1;$$
$$\frac{\partial f_1(\mathbf{x})}{\partial x_j} = 0, j = 2, 3, \cdots, n;$$
$$\frac{\partial f_2(\mathbf{x})}{\partial x_j} = \frac{\partial H(x_1)}{\partial x_j} + \frac{\partial G(\mathbf{x})}{\partial x_j} \cdot S(x_1) + G(\mathbf{x}) \cdot \frac{\partial S(x_1)}{\partial x_j}, j = 1, 2, \cdots, n;$$

$$\frac{\partial H(x_1)}{\partial x_1} = -2x_1;$$
$$\frac{\partial H(x_1)}{\partial x_j} = 0, j = 2, 3, \cdots, n;$$
$$\frac{\partial G(\mathbf{x})}{\partial x_1} = 0;$$
$$\frac{\partial G(\mathbf{x})}{\partial x_j} = 2x_j + 40\pi \sin(4\pi x_j), j = 2, 3, \cdots, n;$$
$$\frac{\partial S(x_1)}{\partial x_1} = -\frac{1}{(0.2 + x_1)^2} + 20x_1;$$
$$\frac{\partial S(x_1)}{\partial x_j} = 0, j = 2, 3, \cdots, n. \tag{A-4}$$

A.3 Test Problem 3 (TP3)

This is the third bi-objective test problem introduced in [63].

minimize $f_1(\mathbf{x}) = x_1$;
minimize $f_2(\mathbf{x}) = H(x_2) \cdot (G(\mathbf{x}) + S(x_1))$;
subject to $0 \leq x_1, x_2 \leq 1$; $-1 \leq x_j \leq 1$, $j = 3, 4, \cdots, n$;
where

$$H(x_2) = 2 - 0.8 \exp\left(-\left(\frac{x_2 - 0.35}{0.25}\right)^2\right) - \exp\left(-\left(\frac{x_2 - 0.85}{0.03}\right)^2\right);$$
$$G(\mathbf{x}) = \sum_{j=3}^{n} 50x_j^2;$$
$$S(x_1) = 1 - \sqrt{x_1}. \tag{A-5}$$

The second objective function of this test problem is not differentiable at $x_1 = 0$. We, therefore, consider $\epsilon \leq x_1 \leq 1$, where ϵ is a very small positive value. Throughout this work, we consider $\epsilon = 1.0 \times 10^{-06}$. For this test problem, following holds true.

$$\frac{\partial f_1(\mathbf{x})}{\partial x_1} = 1;$$
$$\frac{\partial f_1(\mathbf{x})}{\partial x_j} = 0, j = 2, 3, \cdots, n;$$

$$\frac{\partial f_2(\mathbf{x})}{\partial x_j} = \frac{\partial H(x_2)}{\partial x_j} \cdot (G(\mathbf{x}) + S(x_1)) + H(x_2) \cdot \left(\frac{\partial G(\mathbf{x})}{\partial x_j} + \frac{\partial S(x_1)}{\partial x_j} \right), j = 1, 2, \cdots, n;$$

$$\frac{\partial H(x_2)}{\partial x_2} = \frac{128}{5}(x_2 - 0.35) \exp\left(- \left(\frac{x_2 - 0.35}{0.25} \right)^2 \right) + \frac{20000}{9}(x_2 - 0.85) \exp\left(- \left(\frac{x_2 - 0.85}{0.03} \right)^2 \right);$$

$$\frac{\partial H(x_2)}{\partial x_j} = 0, j = 1, 3, 4, \cdots, n;$$

$$\frac{\partial G(\mathbf{x})}{\partial x_j} = 0, j = 1, 2;$$

$$\frac{\partial G(\mathbf{x})}{\partial x_j} = 100 x_j, j = 3, 4, \cdots, n;$$

$$\frac{\partial S(x_1)}{\partial x_1} = -\frac{1}{2\sqrt{x_1}};$$

$$\frac{\partial S(x_1)}{\partial x_j} = 0, j = 2, 3, \cdots, n. \tag{A-6}$$

A.4 Test Problem 4 (TP4)

This is the forth bi-objective test problem introduced in [63].

$$\begin{aligned}
&\text{minimize } f_1(\mathbf{x}) = x_1; \\
&\text{minimize } f_2(\mathbf{x}) = H(x_1, x_2) \cdot (G(\mathbf{x}) + S(x_1)); \\
&\text{subject to } 0 \le x_1 \le 1;\ -0.15 \le x_2 \le 1;\ -1 \le x_j \le 1,\ j = 3, 4, \cdots, n; \\
&\quad \text{where} \\
&\qquad H(x_1, x_2) = 2 - x_1 - 0.8 \exp\left(- \left(\frac{x_1 + x_2 - 0.35}{0.25} \right)^2 \right) - \exp\left(- \left(\frac{x_1 + x_2 - 0.85}{0.03} \right)^2 \right); \\
&\qquad G(\mathbf{x}) = \sum_{j=3}^{n} 50 x_j^2; \\
&\qquad S(x_1) = 1 - \sqrt{x_1}.
\end{aligned} \tag{A-7}$$

The second objective function of this test problem is not differentiable at $x_1 = 0$. We, therefore, consider $\epsilon \le x_1 \le 1$, where ϵ is a very small positive value. Throughout

this work, we consider $\epsilon = 1.0 \times 10^{-06}$. For this test problem, following holds true.

$$
\begin{aligned}
&\frac{\partial f_1(\mathbf{x})}{\partial x_1} = 1;\\
&\frac{\partial f_1(\mathbf{x})}{\partial x_j} = 0, j = 2, 3, \cdots, n;\\
&\frac{\partial f_2(\mathbf{x})}{\partial x_j} = \frac{\partial H(x_1, x_2)}{\partial x_j} \cdot (G(\mathbf{x}) + S(x_1))\\
&\qquad + H(x_1, x_2) \cdot \left(\frac{\partial G(\mathbf{x})}{\partial x_j} + \frac{\partial S(x_1)}{\partial x_j} \right), j = 1, 2, \cdots, n;\\
&\frac{\partial H(x_1, x_2)}{\partial x_1} = -1 + \frac{128}{5} (x_1 + x_2 - 0.35) \exp\left(- \left(\frac{x_1 + x_2 - 0.35}{0.25} \right)^2 \right)\\
&\qquad + \frac{20000}{9} (x_1 + x_2 - 0.85) \exp\left(- \left(\frac{x_1 + x_2 - 0.85}{0.03} \right)^2 \right)\\
&\frac{\partial H(x_1, x_2)}{\partial x_2} = \frac{128}{5} (x_1 + x_2 - 0.35) \exp\left(- \left(\frac{x_1 + x_2 - 0.35}{0.25} \right)^2 \right)\\
&\qquad + \frac{20000}{9} (x_1 + x_2 - 0.85) \exp\left(- \left(\frac{x_1 + x_2 - 0.85}{0.03} \right)^2 \right)\\
&\frac{\partial H(x_1, x_2)}{\partial x_j} = 0, j = 3, 4, \cdots, n;\\
&\frac{\partial G(\mathbf{x})}{\partial x_j} = 0, j = 1, 2;\\
&\frac{\partial G(\mathbf{x})}{\partial x_j} = 100 x_j, j = 3, 4, \cdots, n;\\
&\frac{\partial S(x_1)}{\partial x_1} = -\frac{1}{2\sqrt{x_1}};\\
&\frac{\partial S(x_1)}{\partial x_j} = 0, j = 2, 3, \cdots, n.
\end{aligned}
\tag{A-8}
$$

A.5 Test Problem 5 (TP5)

This is the first tri-objective test problem introduced in [63].

$$
\begin{aligned}
&\text{minimize } f_1(\mathbf{x}) = x_1;\\
&\text{minimize } f_2(\mathbf{x}) = x_2;\\
&\text{minimize } f_3(\mathbf{x}) = H(x_1, x_2) + G(\mathbf{x}) \cdot S(x_1, x_2);
\end{aligned}
$$

subject to $0 \leq x_1, x_2 \leq 1;\ -1 \leq x_j \leq 1,\ j = 3, 4, \cdots, n;$

where

$$H(x_1, x_2) = 2 - x_1^2 - x_2^2;$$

$$G(\mathbf{x}) = \sum_{j=3}^{n} \left(10 + x_j^2 - 10\cos\left(4\pi x_j\right)\right);$$

$$S(x_1, x_2) = \frac{0.75}{0.2 + x_1} + 10x_1^8 + \frac{0.75}{0.2 + x_2} + 10x_2^8. \tag{A-9}$$

Each of the objective functions of this test problem is differentiable. For this test problem, following holds true.

$$\frac{\partial f_1(\mathbf{x})}{\partial x_1} = 1;$$

$$\frac{\partial f_1(\mathbf{x})}{\partial x_j} = 0, j = 2, 3, \cdots, n;$$

$$\frac{\partial f_2(\mathbf{x})}{\partial x_2} = 1;$$

$$\frac{\partial f_2(\mathbf{x})}{\partial x_j} = 0, j = 1, 3, 4, \cdots, n;$$

$$\frac{\partial f_3(\mathbf{x})}{\partial x_j} = \frac{\partial H(x_1, x_2)}{\partial x_j} + \frac{\partial G(\mathbf{x})}{\partial x_j} \cdot S(x_1, x_2) + G(\mathbf{x}) \cdot \frac{\partial S(x_1, x_2)}{\partial x_j}, j = 1, 2, \cdots, n;$$

$$\frac{\partial H(x_1, x_2)}{\partial x_j} = -2x_j, j = 1, 2;$$

$$\frac{\partial H(x_1, x_2)}{\partial x_j} = 0, j = 3, 4, \cdots, n;$$

$$\frac{\partial G(\mathbf{x})}{\partial x_j} = 0, j = 1, 2;$$

$$\frac{\partial G(\mathbf{x})}{\partial x_j} = 2x_j + 40\pi\sin(4\pi x_j), j = 3, 4, \cdots, n;$$

$$\frac{\partial S(x_1, x_2)}{\partial x_j} = -\frac{0.75}{\left(0.2 + x_j\right)^2} + 80x_j^7, j = 1, 2;$$

$$\frac{\partial S(x_1, x_2)}{\partial x_j} = 0, j = 3, 4, \cdots, n. \tag{A-10}$$

A.6 Test Problem 6 (TP6)

This is the first tri-objective test problem introduced in [63].

$$
\begin{aligned}
&\text{minimize } f_1(\mathbf{x}) = x_1;\\
&\text{minimize } f_2(\mathbf{x}) = x_2;\\
&\text{minimize } f_3(\mathbf{x}) = H\left(x_3\right) \cdot \left(G\left(\mathbf{x}\right) + S\left(x_1, x_2\right)\right);\\
&\text{subject to } 0 \leq x_1, x_2, x_3 \leq 1;\ -1 \leq x_j \leq 1,\ j = 4, 5, \cdots, n;\\
&\quad \text{where}\\
&\qquad H\left(x_3\right) = 2 - 0.8 \exp\left(-\left(\frac{x_3 - 0.35}{0.25}\right)^2\right) - \exp\left(-\left(\frac{x_3 - 0.85}{0.03}\right)^2\right);\\
&\qquad G\left(\mathbf{x}\right) = \sum_{j=4}^{n} \left(10 + x_j^2 - 10\cos\left(4\pi x_j\right)\right);\\
&\qquad S\left(x_1, x_2\right) = 10 - \sqrt{x_1} - \sqrt{x_2}.
\end{aligned}
\tag{A-11}
$$

The third objective function of this test problem is not differentiable at $x_1 = 0$ and at $x_2 = 0$. We, therefore, consider $\epsilon \leq x_1, x_2 \leq 1$, where ϵ is a very small positive value. Throughout this work, we consider $\epsilon = 1.0 \times 10^{-06}$. For this test problem, following holds true.

$$
\begin{aligned}
&\frac{\partial f_1(\mathbf{x})}{\partial x_1} = 1;\\
&\frac{\partial f_1(\mathbf{x})}{\partial x_j} = 0, j = 2, 3, \cdots, n;\\
&\frac{\partial f_2(\mathbf{x})}{\partial x_2} = 1;\\
&\frac{\partial f_2(\mathbf{x})}{\partial x_j} = 0, j = 1, 3, 4, \cdots, n;\\
&\frac{\partial f_3(\mathbf{x})}{\partial x_j} = \frac{\partial H(x_3)}{\partial x_j} \cdot (G(\mathbf{x}) + S(x_1, x_2)) +\\
&\qquad\qquad + H(x_3) \cdot \left(\frac{\partial G(\mathbf{x})}{\partial x_j} + \frac{\partial S(x_1, x_2)}{\partial x_j}\right), j = 1, 2, \cdots, n;\\
&\frac{\partial H(x_3)}{\partial x_j} = 0, j = 1, 2, 4, 5, \cdots, n;\\
&\frac{\partial H(x_3)}{\partial x_3} = \frac{128}{5}(x_3 - 0.35) \exp\left(-\left(\frac{x_3 - 0.35}{0.25}\right)^2\right)
\end{aligned}
$$

$$+\frac{20000}{9}(x_3-0.85)\exp\left(-\left(\frac{x_3-0.85}{0.03}\right)^2\right);$$

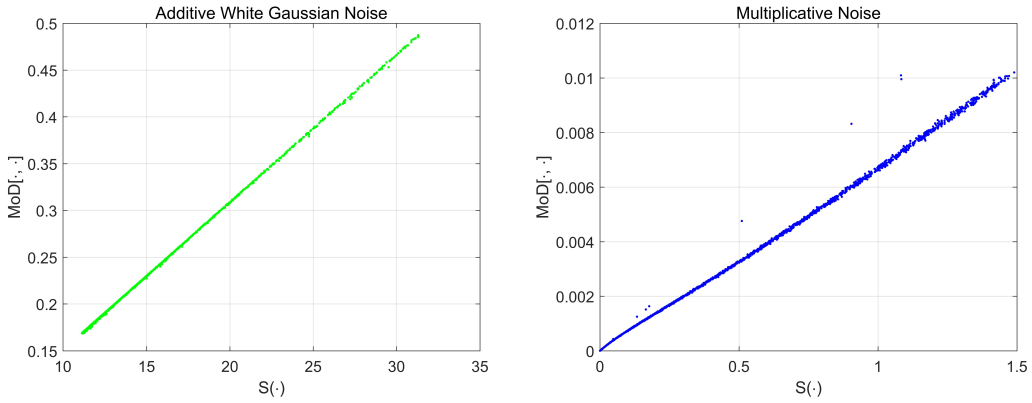
$$\frac{\partial G(\mathbf{x})}{\partial x_j}=0, j=1,2,3;$$

$$\frac{\partial G(\mathbf{x})}{\partial x_j}=2x_j+40\pi\sin(4\pi x_j), j=4,5,\cdots,n;$$

$$\frac{\partial S(x_1,x_2)}{\partial x_j}=-\frac{1}{2\sqrt{x_j}}, j=1,2;$$

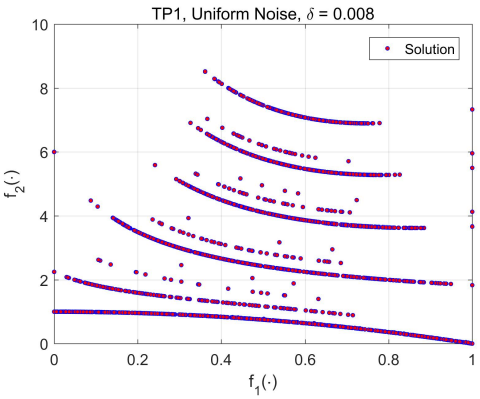
$$\frac{\partial S(x_1,x_2)}{\partial x_j}=0, j=3,4,\cdots,n. \tag{A-12}$$

B Supplementary Figures

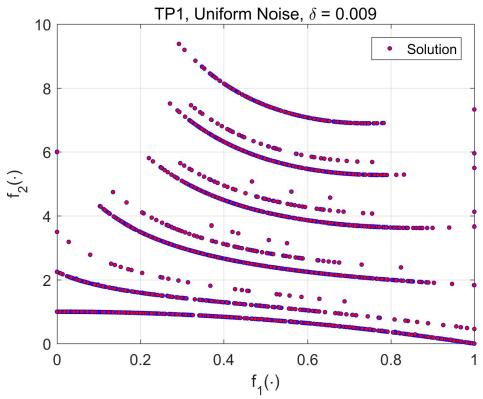


(a) AWGN (b) MN

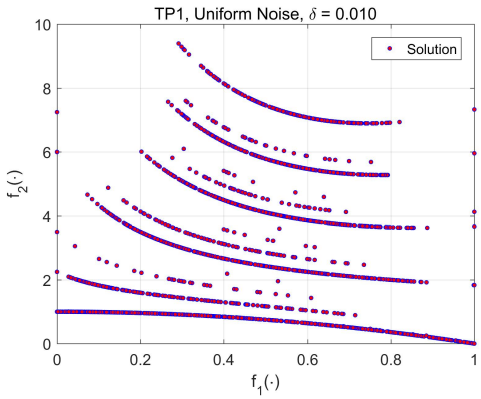
Figure A-1: $\texttt{MoD}[\cdot,\cdot]$-versus-$\mathcal{S}(\cdot)$ plots of the obtained type I robust solutions with $\delta = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$) for (a) AWGN and (b) MN. The search process considered uniform noise.



(a) $\delta = \delta_2 = 0.008$

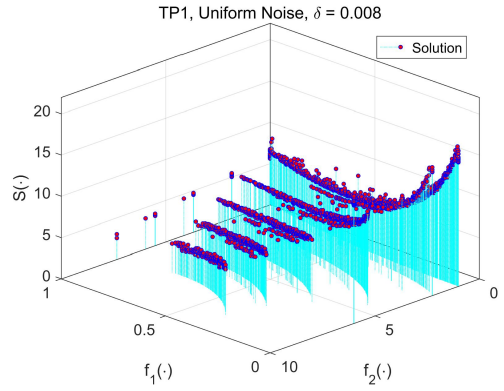


(b) $\delta = \delta_3 = 0.009$

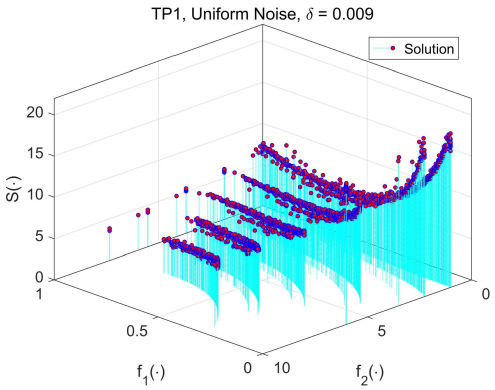


(c) $\delta = \delta_4 = 0.010$

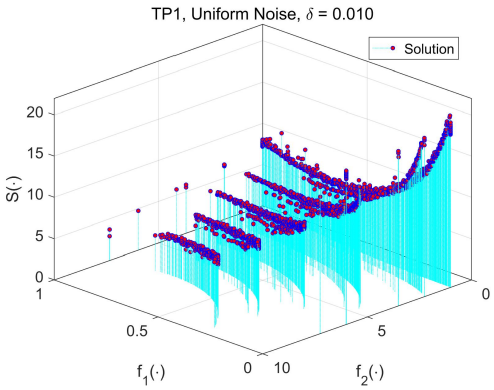
Figure A-2: Scatter plots of the obtained solutions on TP1 using Approach I for uniform noise with (a) $\delta = \delta_2 = 0.008$, (b) $\delta = \delta_3 = 0.009$, and (c) $\delta = \delta_4 = 0.010$.



(a) $\delta = \delta_2 = 0.008$

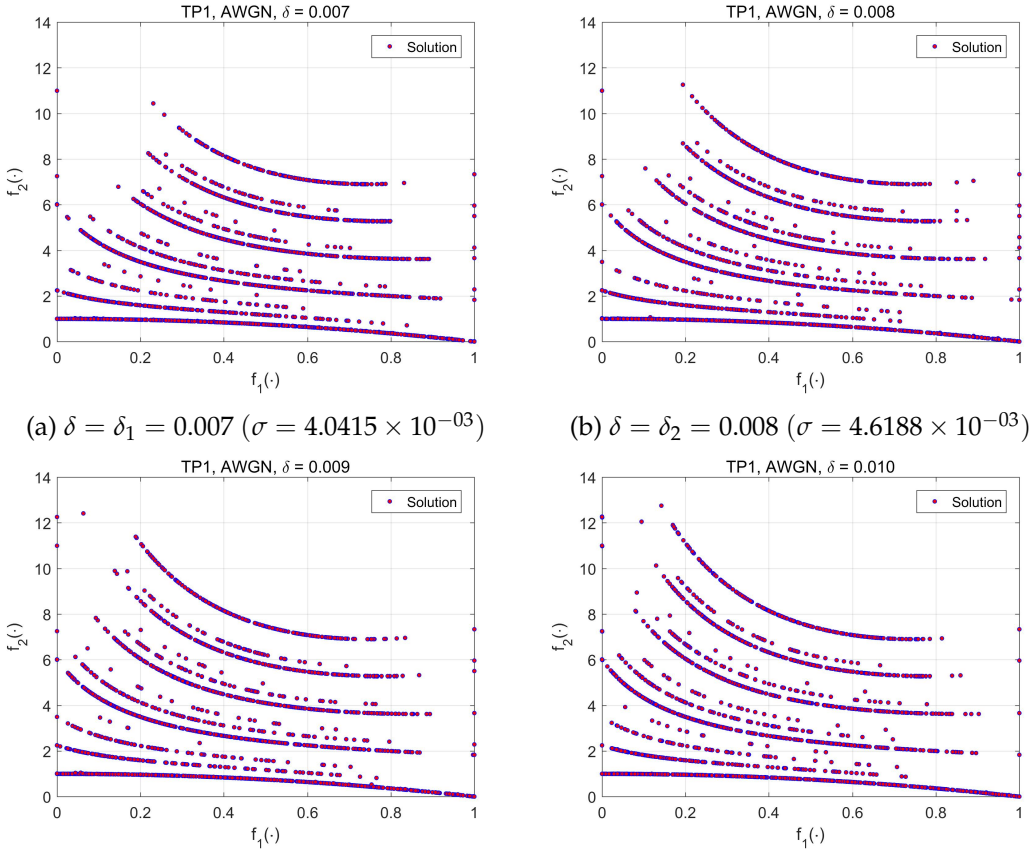


(b) $\delta = \delta_3 = 0.009$



(c) $\delta = \delta_4 = 0.010$

Figure A-3: Three-dimensional stem plots of the obtained solutions on TP1 using Approach I for uniform noise with (a) $\delta = \delta_2 = 0.008$, (b) $\delta = \delta_3 = 0.009$, and (c) $\delta = \delta_4 = 0.010$.



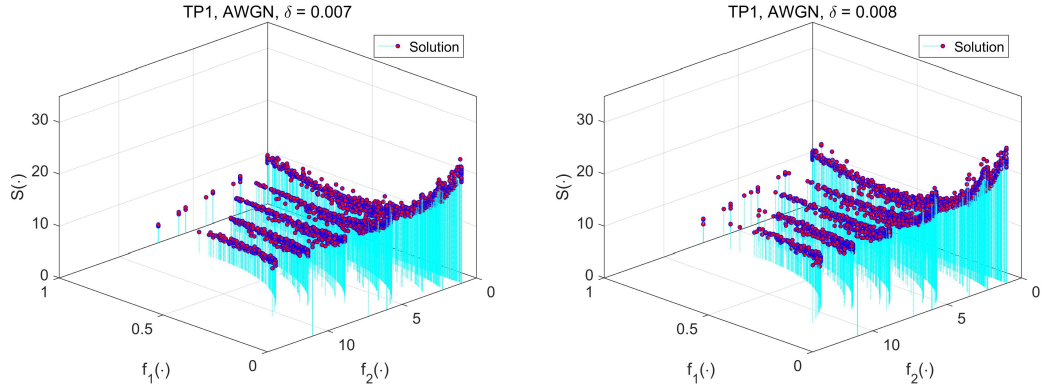
(a) $\delta = \delta_1 = 0.007$ $(\sigma = 4.0415 \times 10^{-03})$

(b) $\delta = \delta_2 = 0.008$ $(\sigma = 4.6188 \times 10^{-03})$

(c) $\delta = \delta_3 = 0.009$ $(\sigma = 5.1962 \times 10^{-03})$

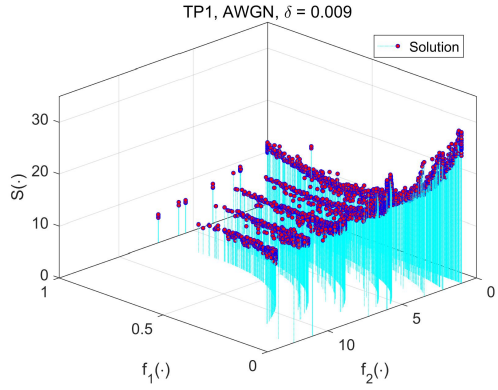
(d) $\delta = \delta_4 = 0.010$ $(\sigma = 5.7735 \times 10^{-03})$

Figure A-4: Scatter plots of the obtained solutions on TP1 using Approach I for AWGN with (a) $\delta = \delta_1 = 0.007$ $(\sigma = 4.0415 \times 10^{-03})$, (b) $\delta = \delta_2 = 0.008$ $(\sigma = 4.6188 \times 10^{-03})$, (c) $\delta = \delta_3 = 0.009$ $(\sigma = 5.1962 \times 10^{-03})$, and (d) $\delta = \delta_4 = 0.010$ $(\sigma = 5.7735 \times 10^{-03})$.

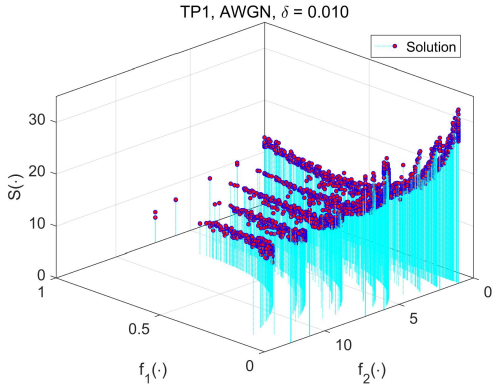


(a) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$)

(b) $\delta = \delta_2 = 0.008$ ($\sigma = 4.6188 \times 10^{-03}$)

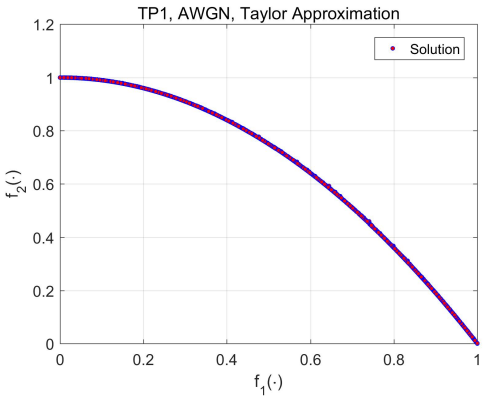


(c) $\delta = \delta_3 = 0.009$ ($\sigma = 5.1962 \times 10^{-03}$)

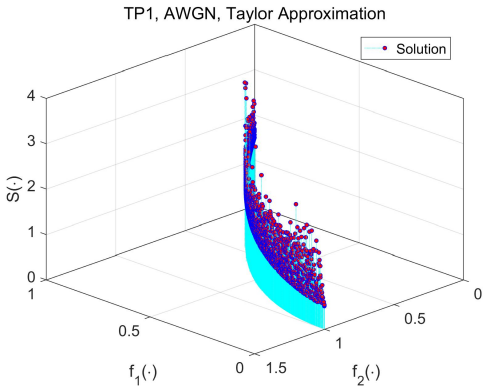


(d) $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$)

Figure A-5: Three-dimensional stem plots of the obtained solutions on TP1 using Approach I for AWGN with (a) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$), (b) $\delta = \delta_2 = 0.008$ ($\sigma = 4.6188 \times 10^{-03}$), (c) $\delta = \delta_3 = 0.009$ ($\sigma = 5.1962 \times 10^{-03}$), and (d) $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$).

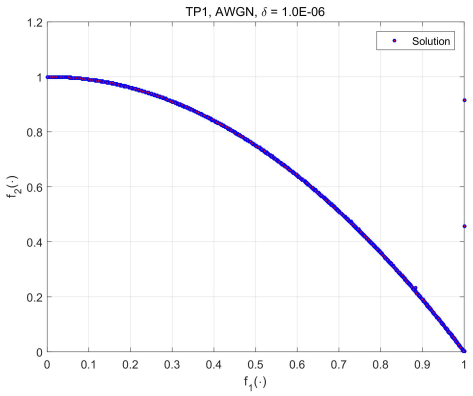


(a) Scatter plot

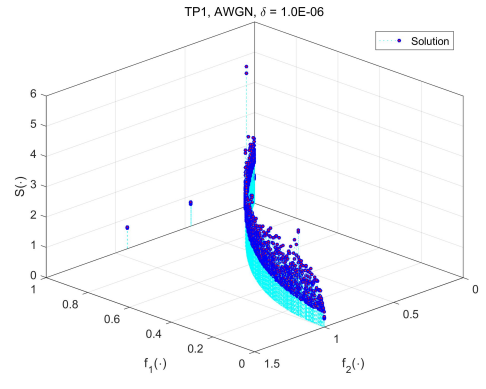


(b) Three-dimensional stem plot

Figure A-6: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP1 using Approach I with Taylor approximation for AWGN.

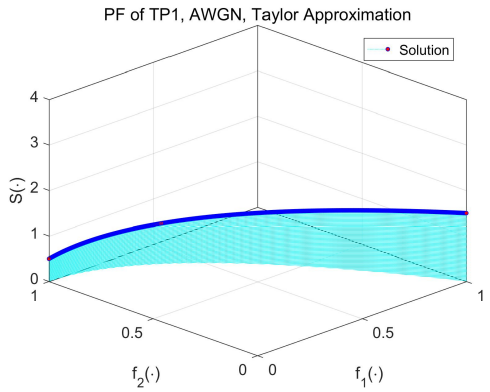


(a) Scatter plot

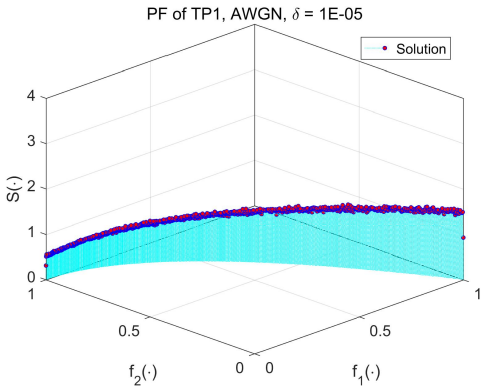


(b) Three-dimensional stem plot

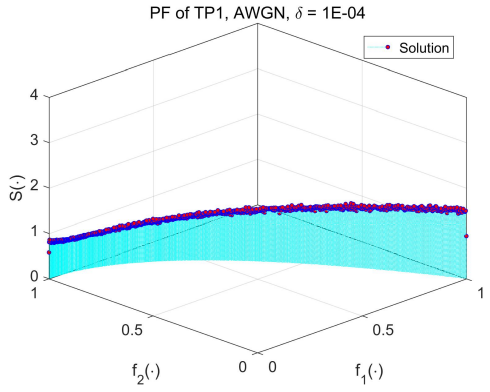
Figure A-7: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP1 using Approach I for AWGN with $\delta = 1.0 \times 10^{-06}$ ($\sigma = 5.7735 \times 10^{-07}$).



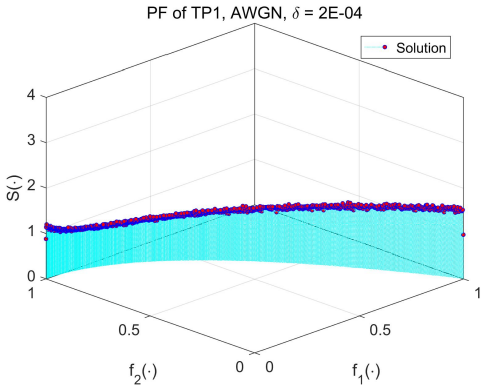
(a) Taylor Approximation



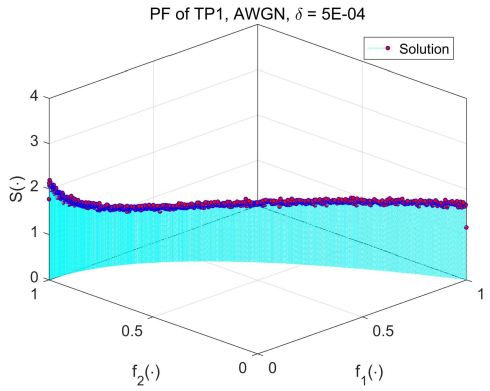
(b) $\delta = 1.0 \times 10^{-05}$ $(\sigma = 5.7735 \times 10^{-06})$



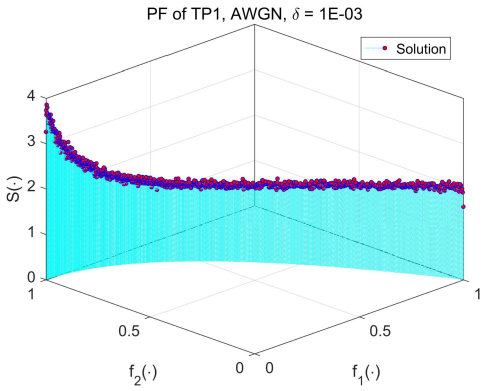
(c) $\delta = 1.0 \times 10^{-04}$ $(\sigma = 5.7735 \times 10^{-05})$



(d) $\delta = 2.0 \times 10^{-04}$ $(\sigma = 1.1547 \times 10^{-04})$



(e) $\delta = 5.0 \times 10^{-04}$ $(\sigma = 2.8868 \times 10^{-04})$



(f) $\delta = 1.0 \times 10^{-03}$ $(\sigma = 5.7735 \times 10^{-04})$

Figure A-8: Three-dimensional stem plots of the solutions corresponding to the PF of TP1 for AWGN with (a) Taylor approximation, (b) $\delta = 1.0 \times 10^{-05}$ $(\sigma = 5.7735 \times 10^{-06})$, (c) $\delta = 1.0 \times 10^{-04}$ $(\sigma = 5.7735 \times 10^{-05})$, (d) $\delta = 2.0 \times 10^{-04}$ $(\sigma = 1.1547 \times 10^{-04})$, (e) $\delta = 5.0 \times 10^{-04}$ $(\sigma = 2.8868 \times 10^{-04})$, and (f) $\delta = 1.0 \times 10^{-03}$ $(\sigma = 5.7735 \times 10^{-04})$.

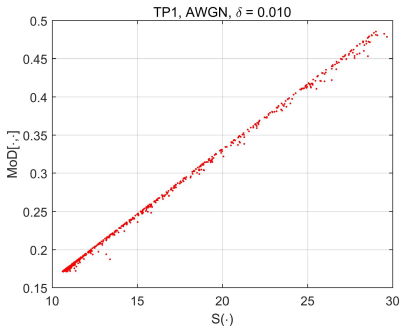
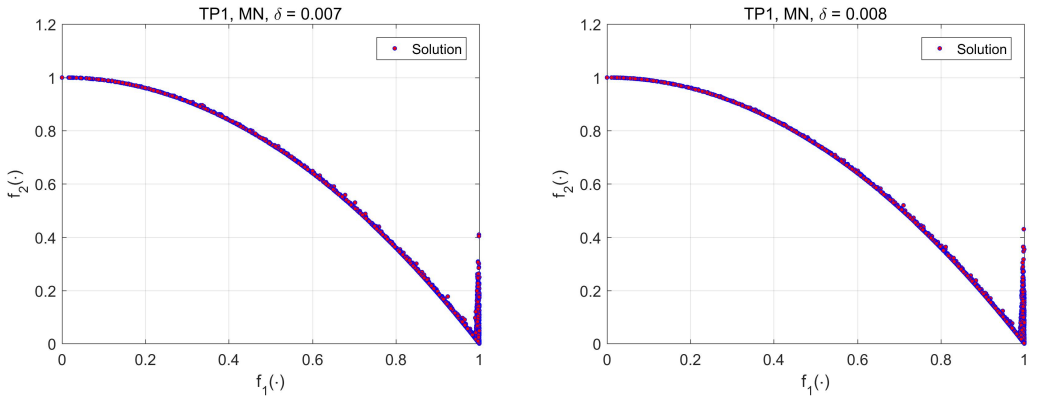


Figure A-9: $\texttt{MoD}[\cdot,\cdot]$-versus-$\mathcal{S}(\cdot)$ plot of the obtained solutions for AWGN on TP1 with $\delta = \delta_4 = 0.010$.



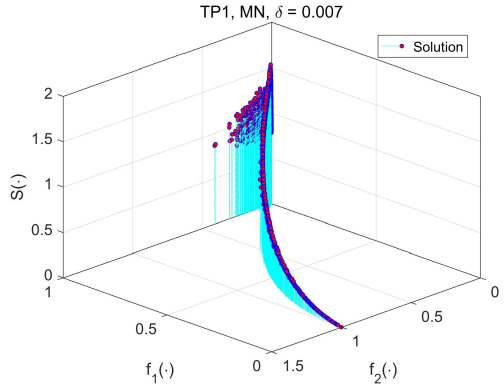
(a) $\delta = \delta_1 = 0.007$ $(\sigma = 4.0415 \times 10^{-03})$

(b) $\delta = \delta_2 = 0.008$ $(\sigma = 4.6188 \times 10^{-03})$

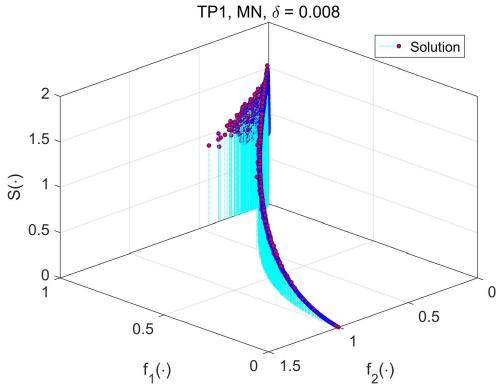
(c) $\delta = \delta_3 = 0.009$ $(\sigma = 5.1962 \times 10^{-03})$

(d) $\delta = \delta_4 = 0.010$ $(\sigma = 5.7735 \times 10^{-03})$

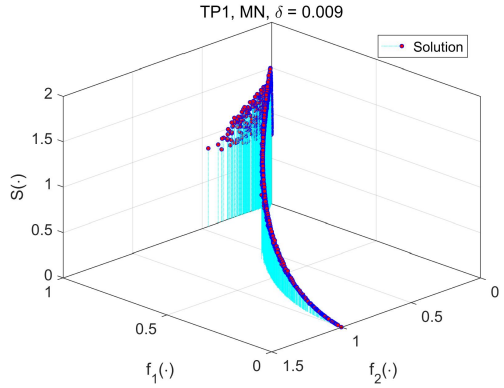
Figure A-10: Scatter plots of the obtained solutions on TP1 using Approach I for MN with (a) $\delta = \delta_1 = 0.007$ $(\sigma = 4.0415 \times 10^{-03})$, (b) $\delta = \delta_2 = 0.008$ $(\sigma = 4.6188 \times 10^{-03})$, (c) $\delta = \delta_3 = 0.009$ $(\sigma = 5.1962 \times 10^{-03})$, and (d) $\delta = \delta_4 = 0.010$ $(\sigma = 5.7735 \times 10^{-03})$.



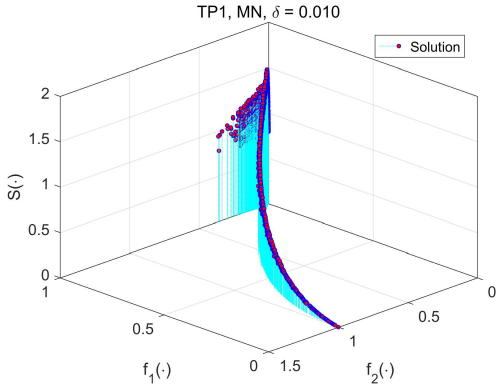
(a) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$)



(b) $\delta = \delta_2 = 0.008$ ($\sigma = 4.6188 \times 10^{-03}$)

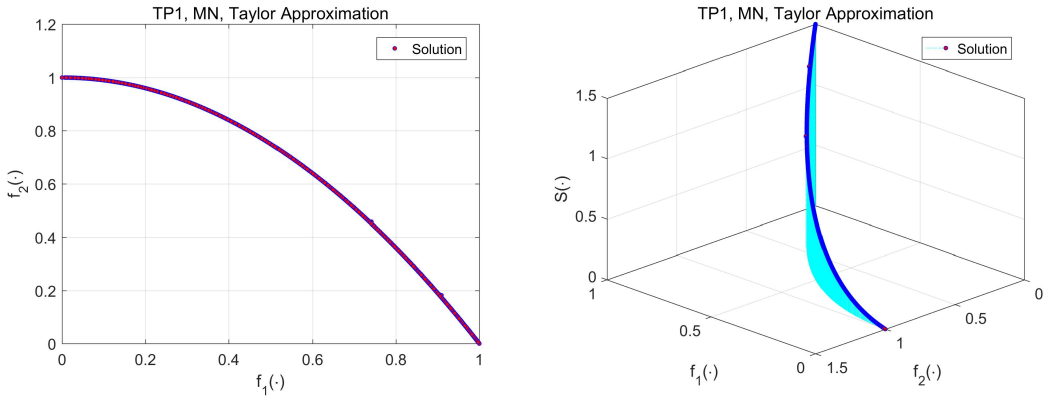


(c) $\delta = \delta_3 = 0.009$ ($\sigma = 5.1962 \times 10^{-03}$)



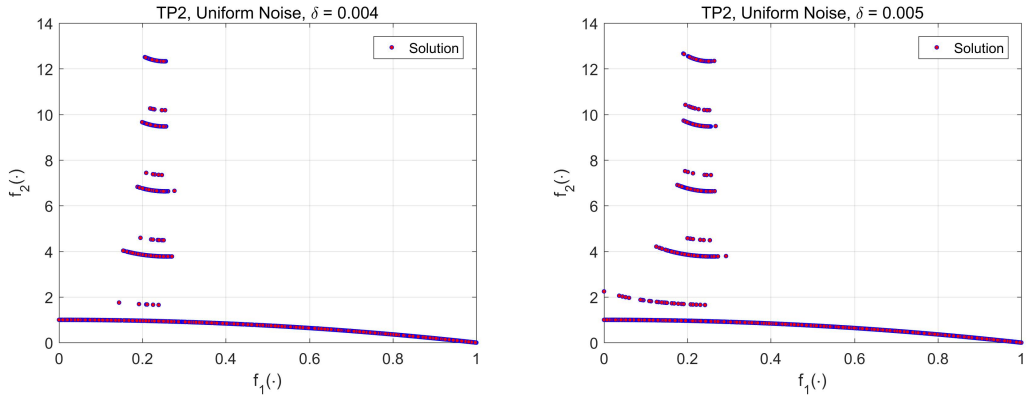
(d) $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$)

Figure A-11: Three-dimensional stem plots of the obtained solutions on TP1 using Approach I for MN with (a) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$), (b) $\delta = \delta_2 = 0.008$ ($\sigma = 4.6188 \times 10^{-03}$), (c) $\delta = \delta_3 = 0.009$ ($\sigma = 5.1962 \times 10^{-03}$), and (d) $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$).



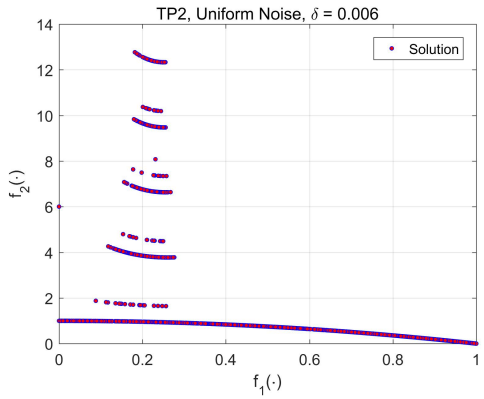
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-12: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP1 using Approach I with Taylor approximation for MN.

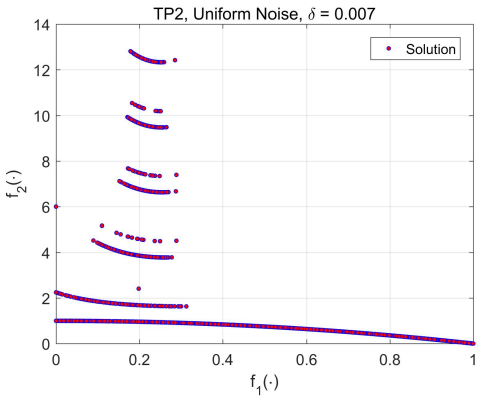


(a) $\delta = \delta_5 = 0.004$

(b) $\delta = \delta_6 = 0.005$

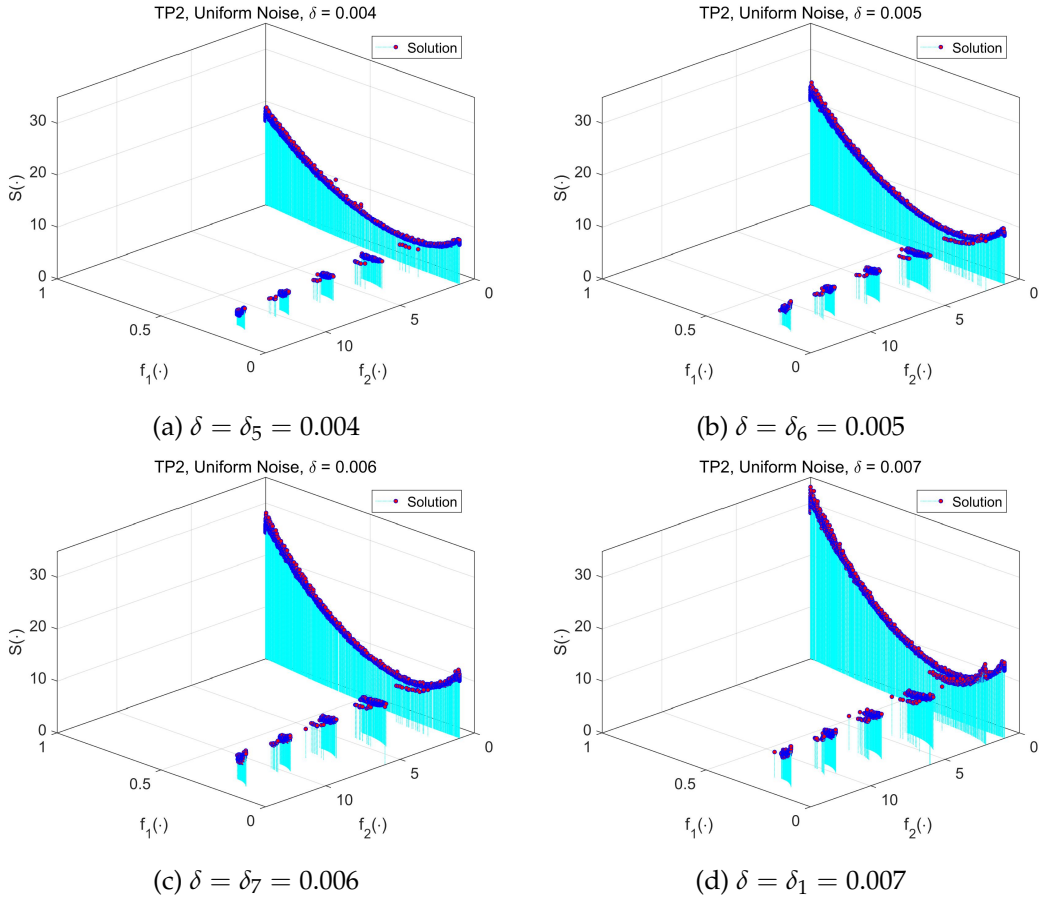


(c) $\delta = \delta_7 = 0.006$



(d) $\delta = \delta_1 = 0.007$

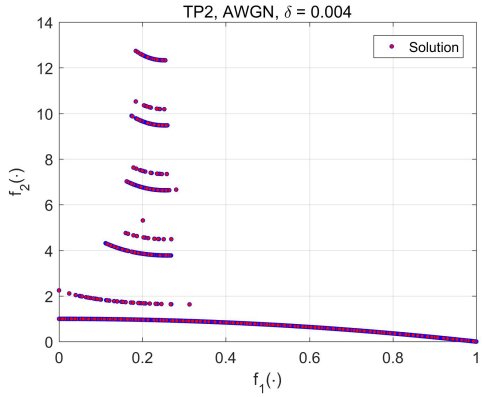
Figure A-13: Scatter plots of the obtained solutions on TP2 using Approach I for uniform noise with (a) $\delta = \delta_5 = 0.004$, (b) $\delta = \delta_6 = 0.005$, (c) $\delta = \delta_7 = 0.006$, and (d) $\delta = \delta_1 = 0.007$.



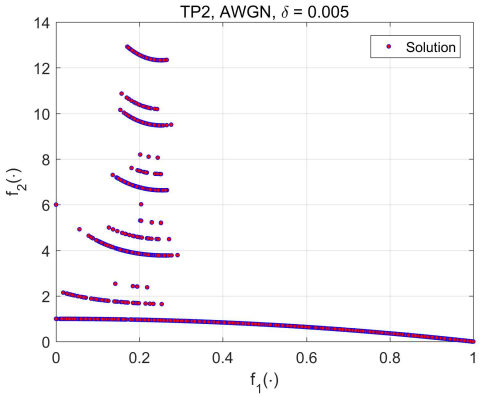
(a) $\delta = \delta_5 = 0.004$ (b) $\delta = \delta_6 = 0.005$

(c) $\delta = \delta_7 = 0.006$ (d) $\delta = \delta_1 = 0.007$

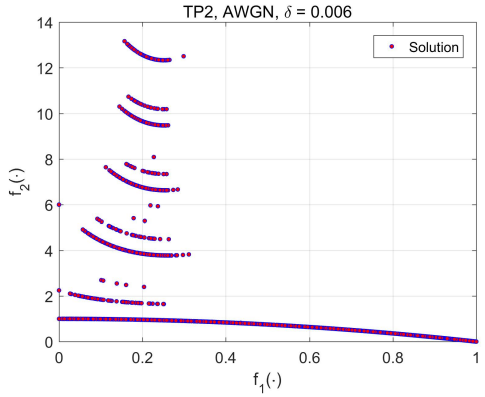
Figure A-14: Three-dimensional stem plots of the obtained solutions on TP2 using Approach I for uniform noise with (a) $\delta = \delta_5 = 0.004$, (b) $\delta = \delta_6 = 0.005$, (c) $\delta = \delta_7 = 0.006$, and (d) $\delta = \delta_1 = 0.007$.



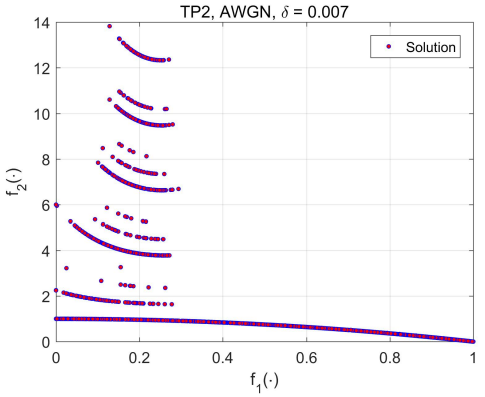
(a) $\delta = \delta_5 = 0.004$ ($\sigma = 2.3094 \times 10^{-03}$)



(b) $\delta = \delta_6 = 0.005$ ($\sigma = 2.8868 \times 10^{-03}$)

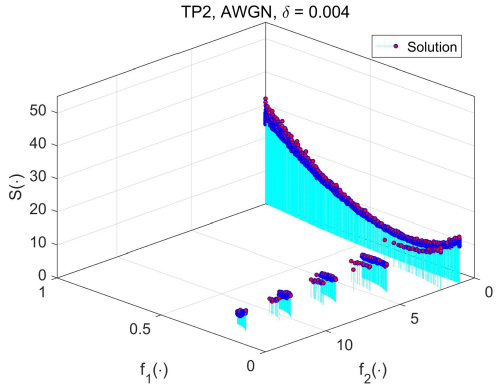


(c) $\delta = \delta_7 = 0.006$ ($\sigma = 3.4641 \times 10^{-03}$)

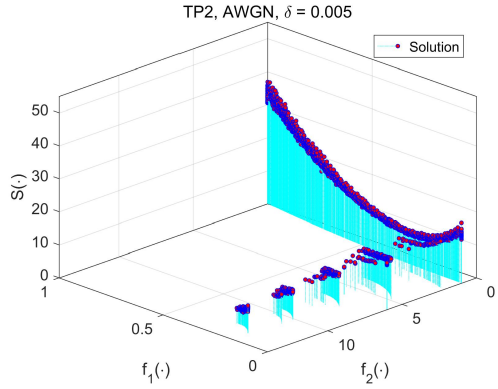


(d) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$)

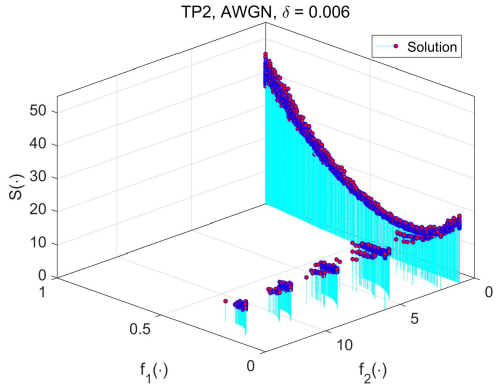
Figure A-15: Scatter plots of the obtained solutions on TP2 using Approach I for AWGN with (a) $\delta = \delta_5 = 0.004$ ($\sigma = 2.3094 \times 10^{-03}$), (b) $\delta = \delta_6 = 0.005$ ($\sigma = 2.8868 \times 10^{-03}$), (c) $\delta = \delta_7 = 0.006$ ($\sigma = 3.4641 \times 10^{-03}$), and (d) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$).



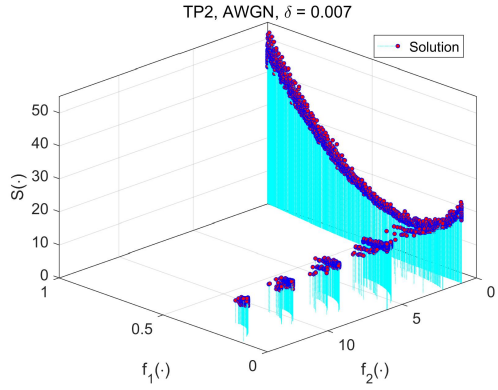
(a) $\delta = \delta_5 = 0.004$ $(\sigma = 2.3094 \times 10^{-03})$



(b) $\delta = \delta_6 = 0.005$ $(\sigma = 2.8868 \times 10^{-03})$

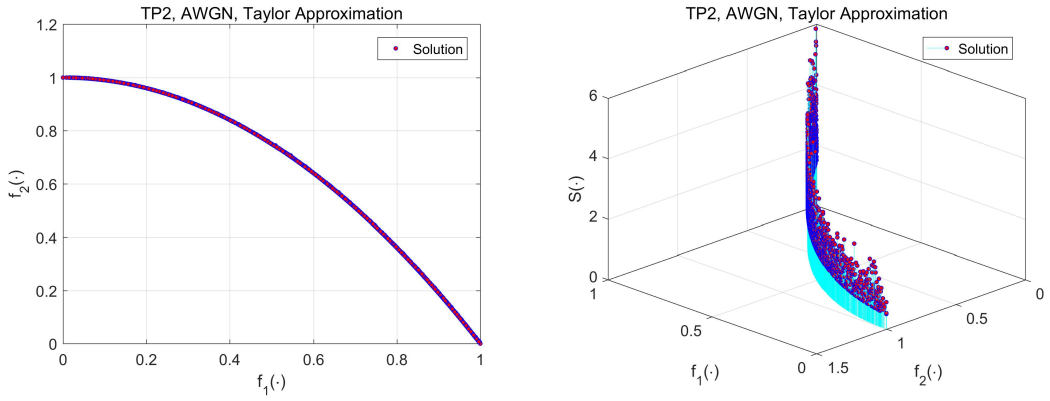


(c) $\delta = \delta_7 = 0.006$ $(\sigma = 3.4641 \times 10^{-03})$



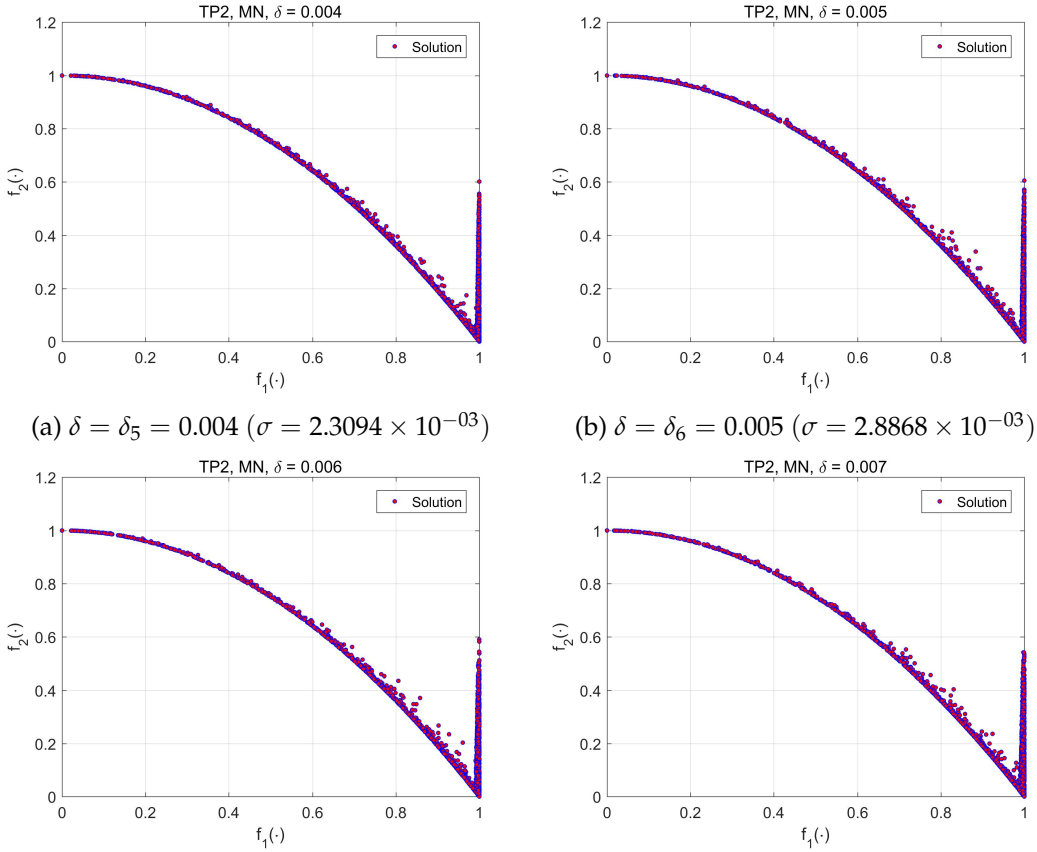
(d) $\delta = \delta_1 = 0.007$ $(\sigma = 4.0415 \times 10^{-03})$

Figure A-16: Three-dimensional stem plots of the obtained solutions on TP2 using Approach I for AWGN with (a) $\delta = \delta_5 = 0.004$ $(\sigma = 2.3094 \times 10^{-03})$, (b) $\delta = \delta_6 = 0.005$ $(\sigma = 2.8868 \times 10^{-03})$, (c) $\delta = \delta_7 = 0.006$ $(\sigma = 3.4641 \times 10^{-03})$, and (d) $\delta = \delta_1 = 0.007$ $(\sigma = 4.0415 \times 10^{-03})$.



(a) Scatter plot (b) Three-dimensional stem plot

Figure A-17: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP2 using Approach I with Taylor approximation for AWGN.



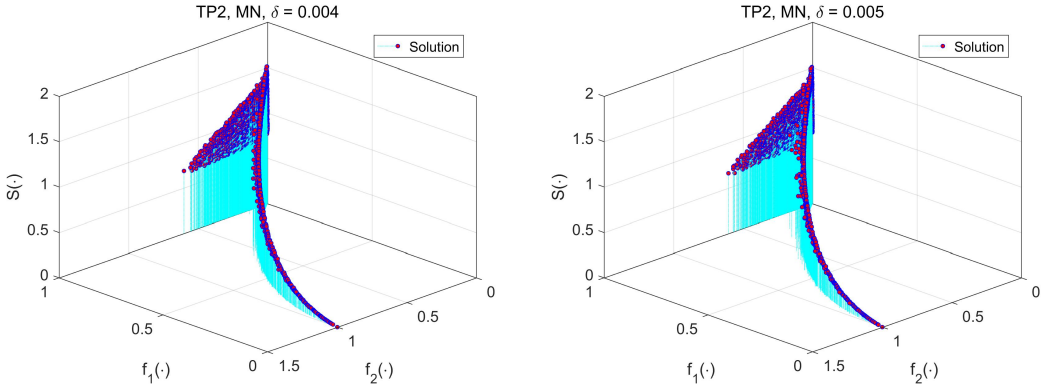
(a) $\delta = \delta_5 = 0.004$ ($\sigma = 2.3094 \times 10^{-03}$)

(b) $\delta = \delta_6 = 0.005$ ($\sigma = 2.8868 \times 10^{-03}$)

(c) $\delta = \delta_7 = 0.006$ ($\sigma = 3.4641 \times 10^{-03}$)

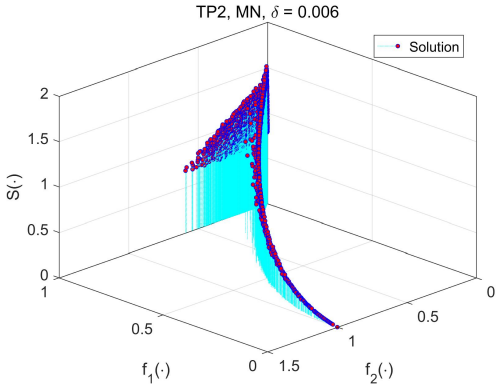
(d) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$)

Figure A-18: Scatter plots of the obtained solutions on TP2 using Approach I for MN with (a) $\delta = \delta_5 = 0.004$ ($\sigma = 2.3094 \times 10^{-03}$), (b) $\delta = \delta_6 = 0.005$ ($\sigma = 2.8868 \times 10^{-03}$), (c) $\delta = \delta_7 = 0.006$ ($\sigma = 3.4641 \times 10^{-03}$), and (d) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$).

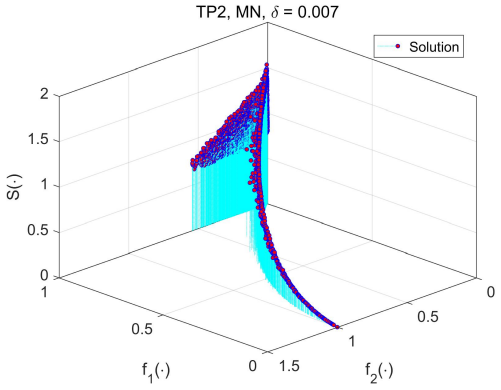


(a) $\delta = \delta_5 = 0.004$ ($\sigma = 2.3094 \times 10^{-03}$)

(b) $\delta = \delta_6 = 0.005$ ($\sigma = 2.8868 \times 10^{-03}$)

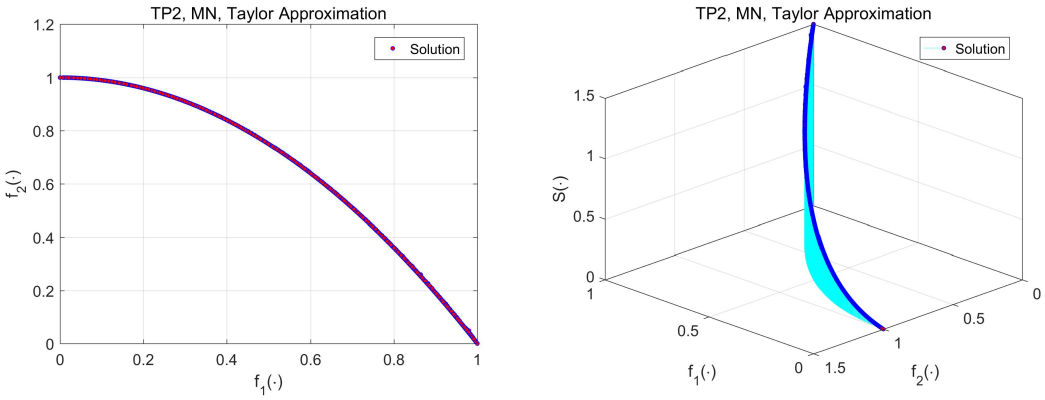


(c) $\delta = \delta_7 = 0.006$ ($\sigma = 3.4641 \times 10^{-03}$)



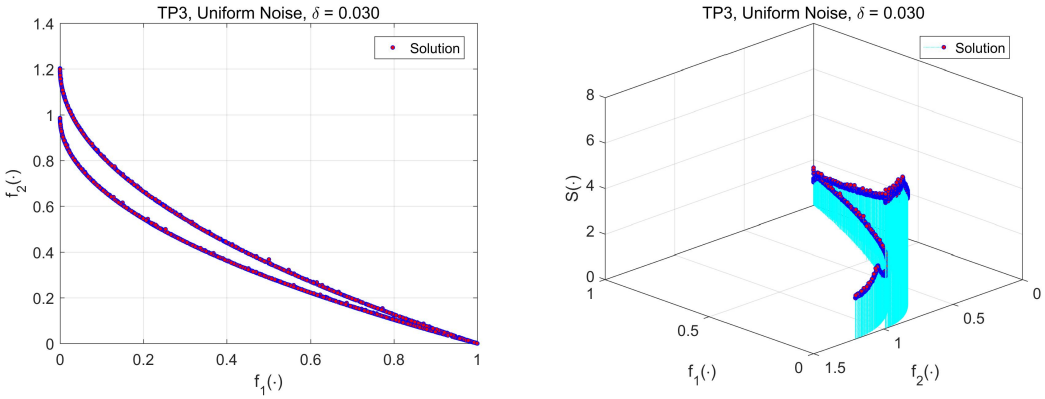
(d) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$)

Figure A-19: Three-dimensional stem plots of the obtained solutions on TP2 using Approach I for MN with (a) $\delta = \delta_5 = 0.004$ ($\sigma = 2.3094 \times 10^{-03}$), (b) $\delta = \delta_6 = 0.005$ ($\sigma = 2.8868 \times 10^{-03}$), (c) $\delta = \delta_7 = 0.006$ ($\sigma = 3.4641 \times 10^{-03}$), and (d) $\delta = \delta_1 = 0.007$ ($\sigma = 4.0415 \times 10^{-03}$).



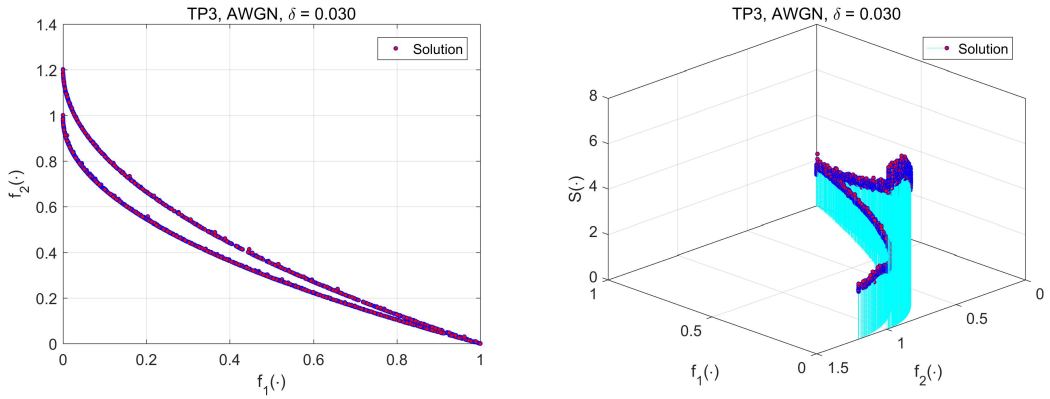
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-20: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP2 using Approach I with Taylor approximation for MN.



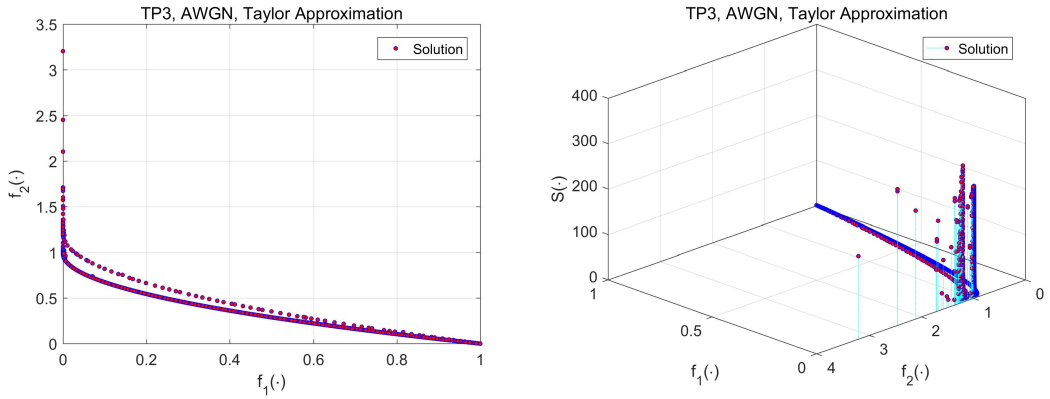
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-21: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP3 using Approach I for uniform noise with $\delta = \delta_8 = 0.030$.



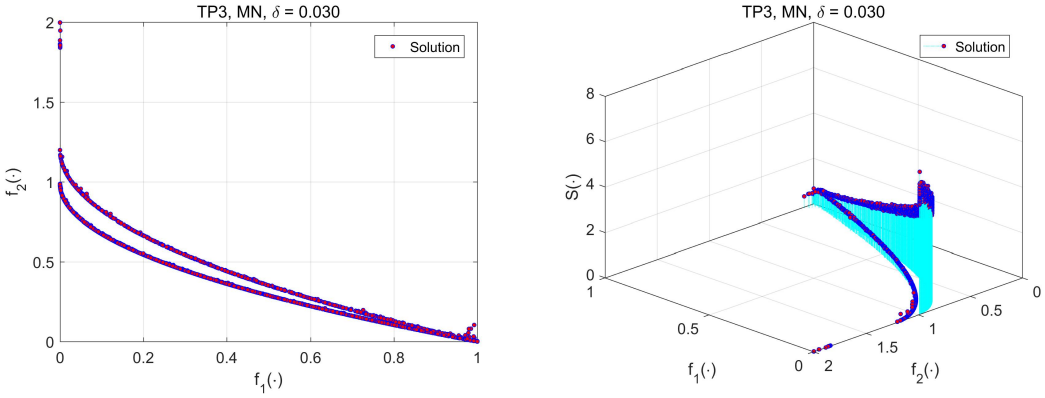
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-22: (a) Scatter plots and (b) three-dimensional stem plots of the obtained solutions on TP3 using Approach I for AWGN with $\delta = \delta_8 = 0.030$ ($\sigma = 1.7321 \times 10^{-02}$).



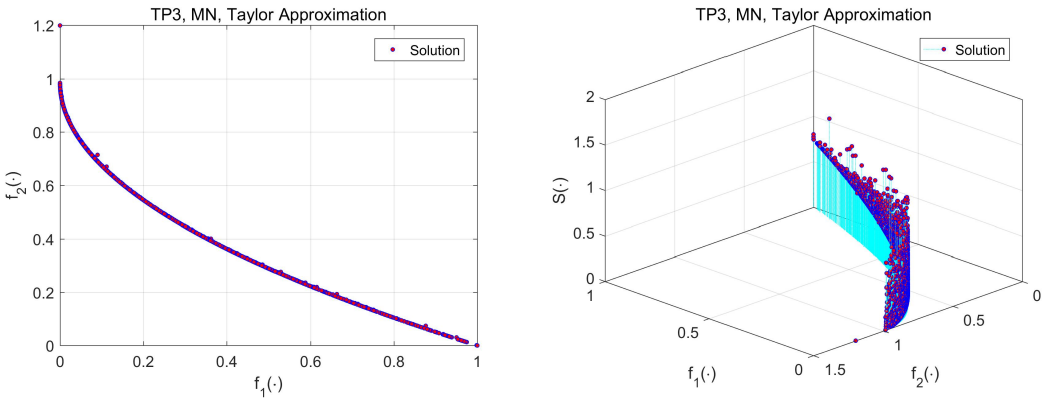
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-23: (a) Scatter plot and (a) three-dimensional stem plot of the obtained solutions on TP3 using Approach I with Taylor approximation for AWGN.



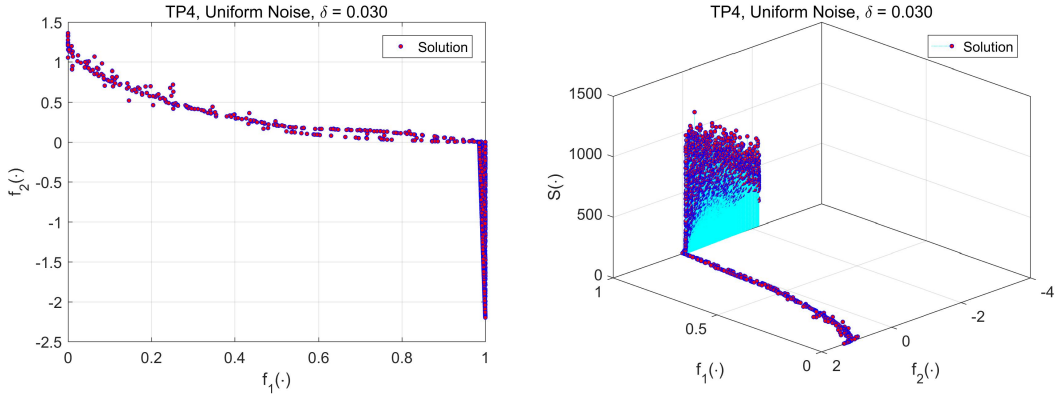
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-24: (a) Scatter plot and (b) three-dimensional stem plots of the obtained solutions on TP3 using Approach I for MN with $\delta = \delta_8 = 0.030$ ($\sigma = 1.7321 \times 10^{-02}$).



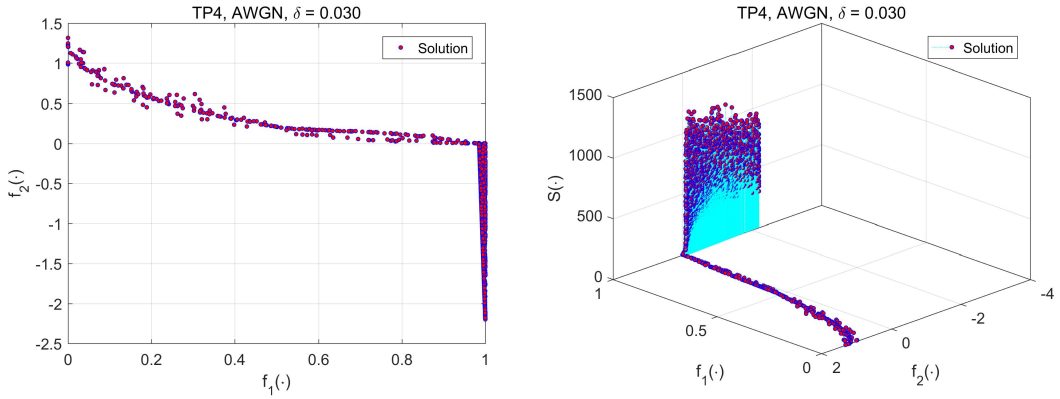
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-25: Scatter plots and three-dimensional stem plots of the obtained solutions on TP3 using Approach I with Taylor approximation for MN.



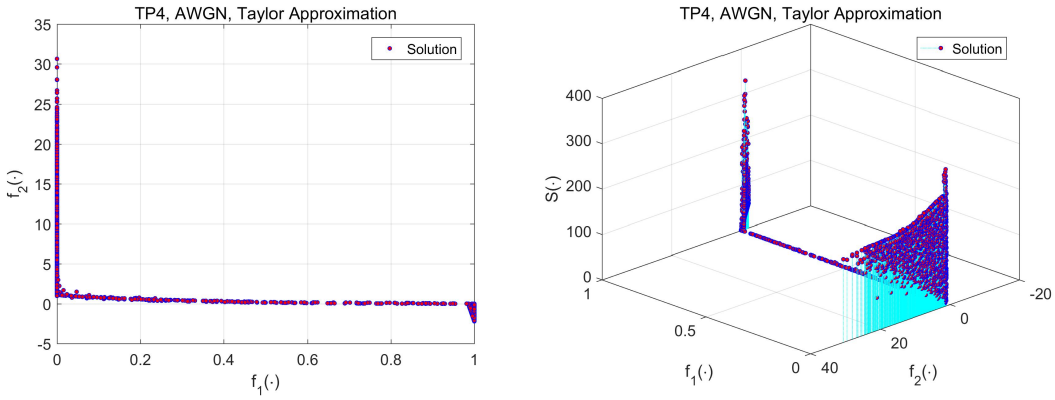
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-26: (a) Scatter plot and (b) three-dimensional stem plots of the obtained solutions on TP4 using Approach I for uniform noise with $\delta = \delta_8 = 0.030$.



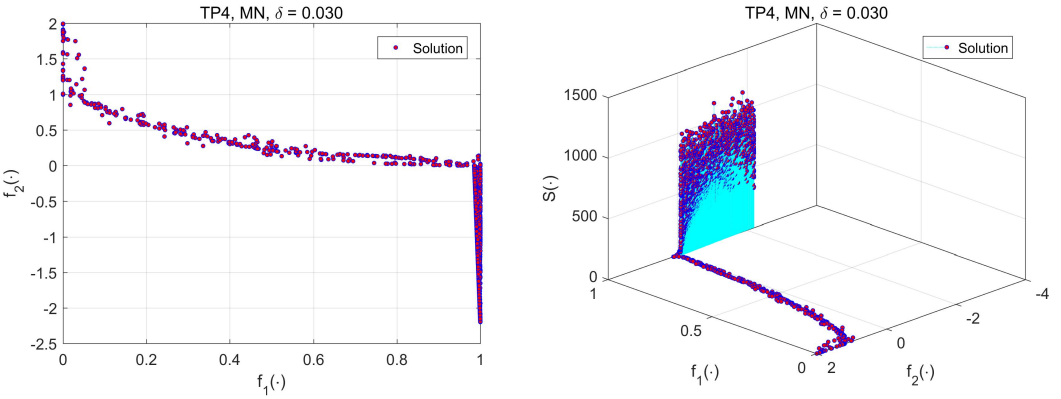
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-27: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP4 using Approach I for AWGN with $\delta = \delta_8 = 0.030$ ($\sigma = 1.7321 \times 10^{-02}$).



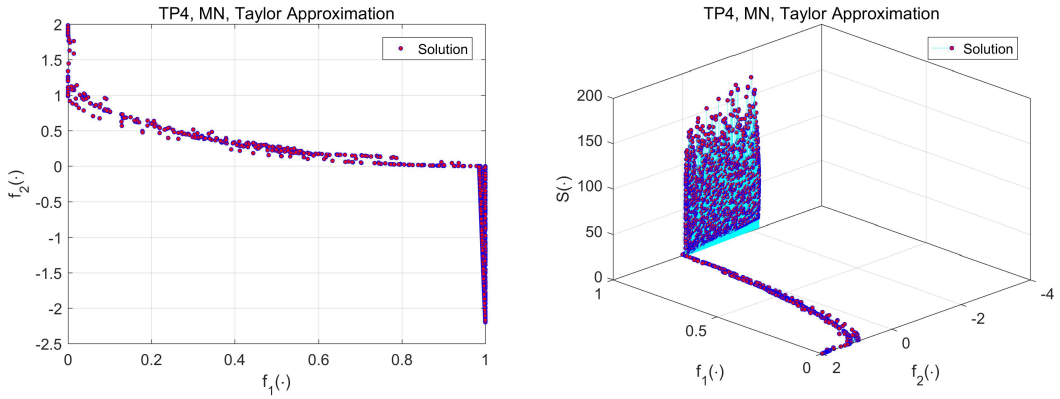
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-28: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP4 using Approach I with Taylor approximation for AWGN.



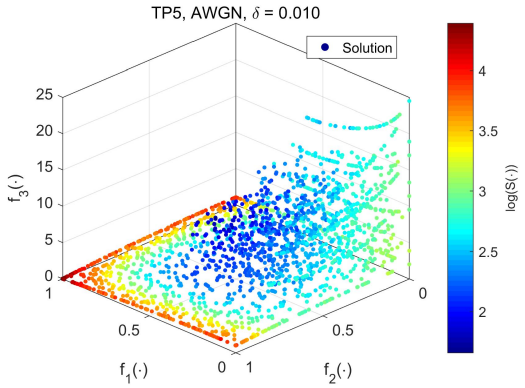
(a) Scatter plot (b) Three-dimensional stem plot

Figure A-29: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP4 using Approach I for MN with $\delta = \delta_8 = 0.030$ ($\sigma = 1.7321 \times 10^{-02}$).

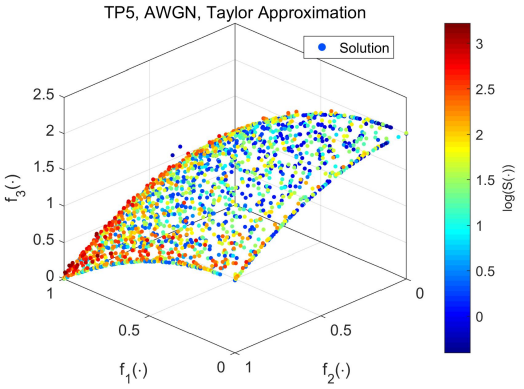


(a) Scatter plot (b) Three-dimensional stem plot

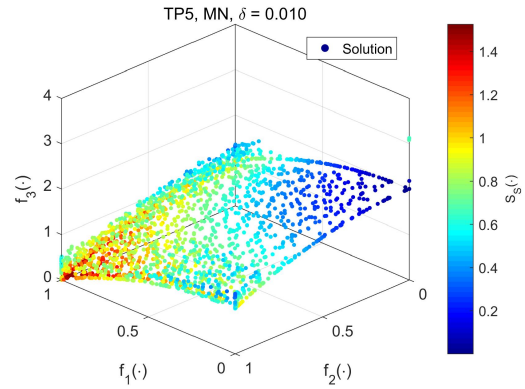
Figure A-30: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP4 using Approach I with Taylor approximation for MN.



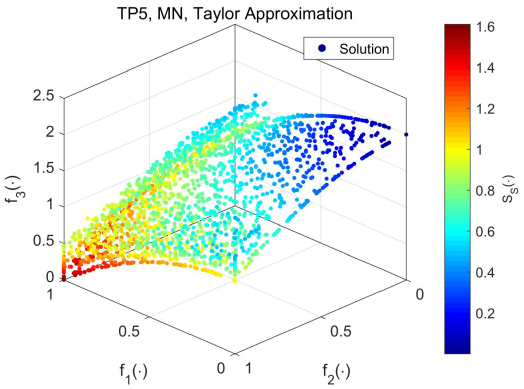
(a) AWGN with $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$)



(b) With approximation for AWGN

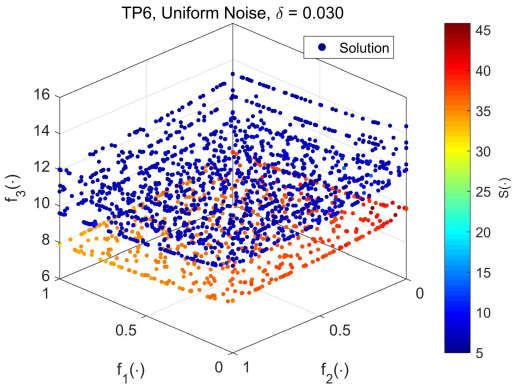


(c) MN with $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$)

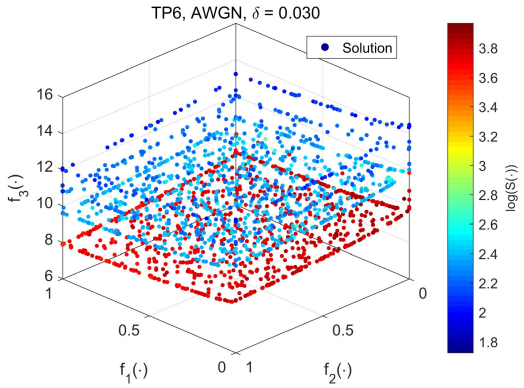


(d) With approximation for MN

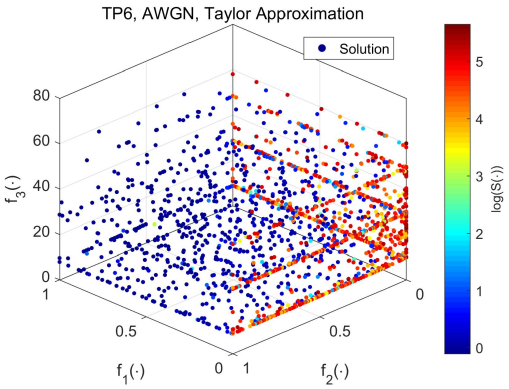
Figure A-31: Four dimensional scatter plots of the obtained solutions on TP5 using Approach I, when (a) AWGN with $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$), (b) Taylor approximation for AWGN, (c) MN with $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$), and (d) Taylor approximation for MN are investigated. The fourth dimension is represented using colors.



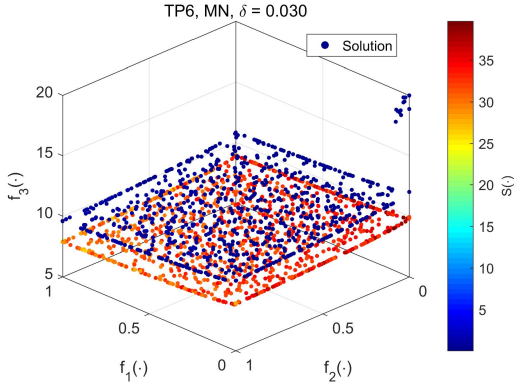
(a) Uniform noise with $\delta = \delta_8 = 0.030$



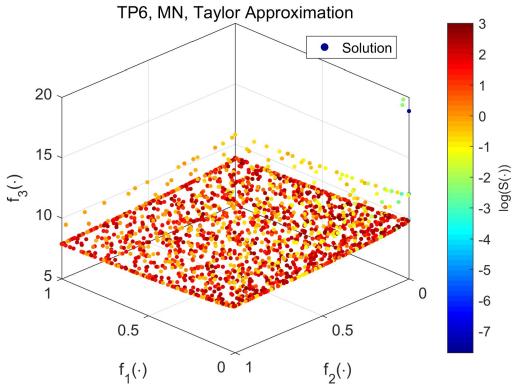
(b) AWGN with $\delta = \delta_8 = 0.030$ ($\sigma = 1.7321 \times 10^{-02}$)



(c) With approximation for AWGN

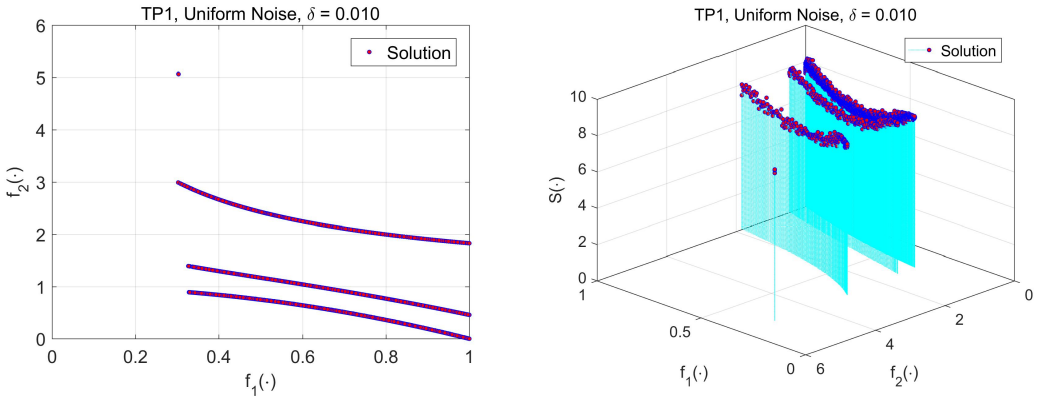


(d) MN with $\delta = \delta_8 = 0.030$ ($\sigma = 1.7321 \times 10^{-02}$)



(e) With approximation for MN

Figure A-32: Four dimensional scatter plots of the obtained solutions on TP6 using Approach I, when (a) uniform noise with $\delta = \delta_8 = 0.030$, (b) AWGN with $\delta = \delta_8 = 0.030$ ($\sigma = 1.7321 \times 10^{-02}$), (c) Taylor approximation for AWGN, (d) MN with $\delta = \delta_8 = 0.030$ ($\sigma = 1.7321 \times 10^{-02}$), and (e) Taylor approximation for MN are investigated. The fourth dimension is represented using colors.



(a) Scatter plot (b) Three-dimensional stem plot

Figure A-33: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP1 using Approach II for uniform noise with $\delta = \delta_4 = 0.010$.

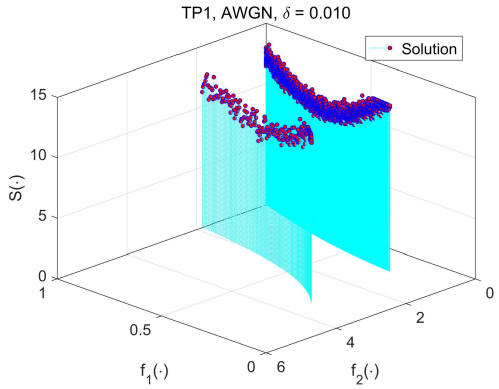
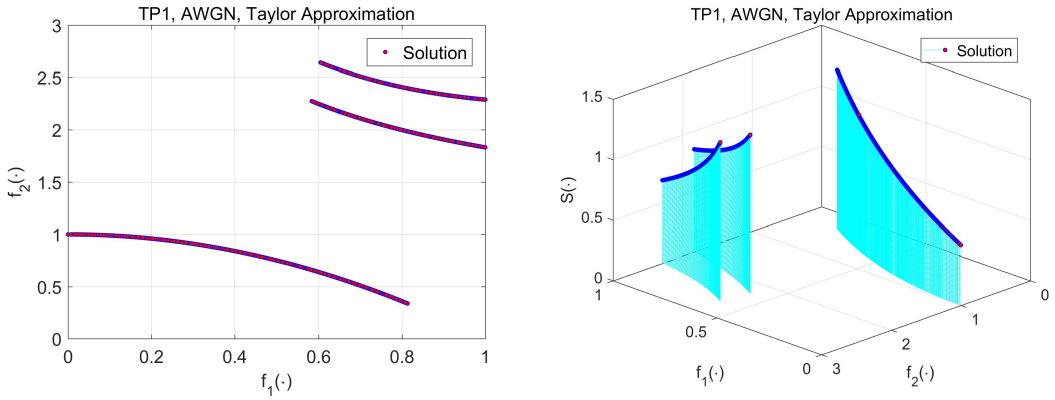


Figure A-34: Three-dimensional stem plot of the obtained solutions on TP1 using Approach II for AWGN with $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$).



(a) Scatter plot (b) Three-dimensional stem plot

Figure A-35: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP1 using Approach II with Taylor approximation for AWGN.

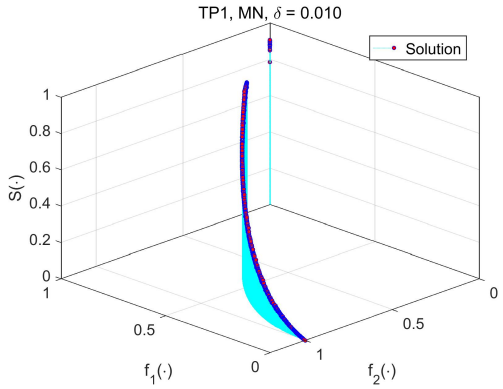
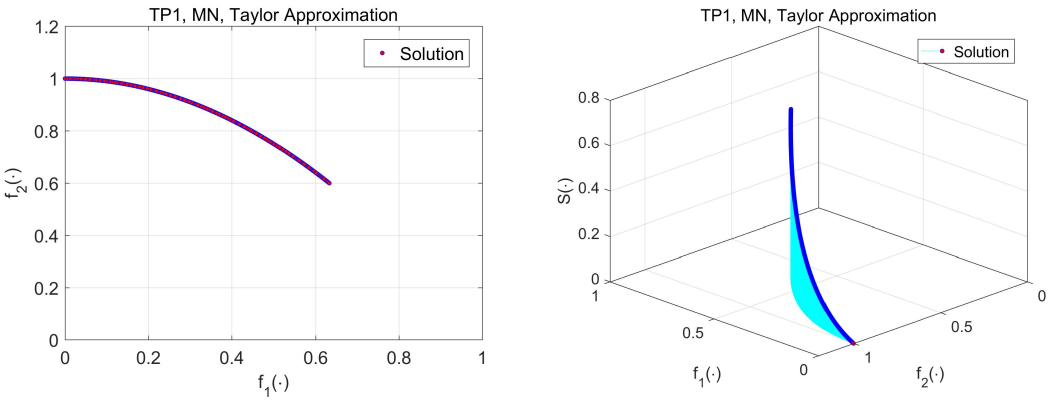


Figure A-36: Three-dimensional stem plot of the obtained solutions on TP1 using Approach II for MN with $\delta = \delta_4 = 0.010$ ($\sigma = 5.7735 \times 10^{-03}$).



(a) Scatter plot

(b) Three-dimensional stem plot

Figure A-37: (a) Scatter plot and (b) three-dimensional stem plot of the obtained solutions on TP1 using Approach II with Taylor approximation for MN.

Bibliography

[1] K. Nag and N. R. Pal, "A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification," *IEEE Transactions on Cybernetics*, vol. 46, no. 2, pp. 499–510, 2016. xvii, xviii, 14, 17, 43, 67

[2] K. Nag and N. R. Pal, "Feature extraction and selection for parsimonious classifiers with multiobjective genetic programming," *IEEE Transactions on Evolutionary Computation*, (revision submitted). xvii, xviii, 14, 41

[3] K. Nag, T. Pal, and N. R. Pal, "Robust consensus: A new measure for multicriteria robust group decision making problems using evolutionary approach," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2014, pp. 384–394. xvii, xix, 15, 65, 67, 69

[4] K. Nag, T. Pal, R. K. Mudi, and N. R. Pal, "Robust mutliobjective optimization with robust consensus," *IEEE Transactions on Fuzzy Systems*, 2018, DOI: 10.1109/TFUZZ.2018.2848261. xvii, xix, 15, 65

[5] K. Nag and N. R. Pal, "Robustness in multiobjective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, (in review). xvii, xix, 16, 93

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, 1995, pp. 1942–1948 vol.4. 2

[7] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142. Paris, France, 1991, pp. 134–142. 2

[8] M. Dorigo, "Optimization, learning and natural algorithms," *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992. 2

[9] http://www.alanturing.net/turing_archive/archive/l/l32/L32-019.html, (accessed on January 17, 2018). 2

[10] N. L. Cramer, "A representation for the adaptive generation of simple sequential programs," in *Proceedings of the First International Conference on Genetic Algorithms*, 1985, pp. 183–187. 3, 8

[11] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press, 1992. 3, 8, 19

[12] J. R. Koza, *Genetic programming II: automatic discovery of reusable programs*. MIT press, 1994. 3, 8

[13] J. R. Koza, F. H. Bennett III, and O. Stiffelman, *Genetic programming as a Darwinian invention machine*. Springer, 1999. 3, 8

[14] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, G. Lanza, and J. Yu, *Genetic programming IV: Routine human-competitive machine intelligence*. Springer Science+ Business Media, 2007, vol. 5. 3

[15] R. S. Rosenberg, *Simulation of genetic populations with biochemical properties*. Ph.D. Thesis, University of Michigan, Ann Harbor, 1967. 3

[16] C. C. Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE computational intelligence magazine*, vol. 1, no. 1, pp. 28–36, 2006. 3

[17] K. Ito, S. Akagi, and M. Nishikawa, "A multiobjective optimization approach to a design problem of heat insulation for thermal distribution piping network systems," *Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 105, no. 2, pp. 206–213, 1983. 3

[18] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications, 1985*. Lawrence Erlbaum Associates Inc., Publishers, 1985. 3

[19] D. Chakraborty and N. R. Pal, "Selecting useful groups of features in a connectionist framework," *Neural Networks, IEEE Transactions on*, vol. 19, no. 3, pp. 381–396, 2008. 5, 6

[20] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997. 6

[21] Y.-C. Chen, N. R. Pal, and I.-F. Chung, "An integrated mechanism for feature selection and fuzzy rule extraction for classification," *Fuzzy Systems, IEEE Transactions on*, vol. 20, no. 4, pp. 683–698, 2012. 6

[22] D. P. Muni, N. R. Pal, and J. Das, "Genetic programming for simultaneous feature selection and classifier design," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 1, pp. 106–117, 2006. 6, 9, 19, 43

[23] N. Pal and K. Chintalapudi, "A connectionist system for feature selection," *Neural Parallel and Scientific Computations*, vol. 5, pp. 359–382, 1997. 6

[24] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002. 6

[25] D. Koller and M. Sahami, "Toward optimal feature selection," Stanford InfoLab, Tech. Rep., 1996. 6

[26] Z. Zhu, Y.-S. Ong, and M. Dash, "Markov blanket-embedded genetic algorithm for gene selection," *Pattern Recognition*, vol. 40, no. 11, pp. 3236–3248, 2007. 6

[27] Z. Zhu, Y.-S. Ong, and J. M. Zurada, "Identification of full and partial class relevant genes," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 7, no. 2, pp. 263–277, 2010. 6, 7

[28] C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. D. Koutsoukos, "Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation," *Journal of Machine Learning Research*, vol. 11, no. Jan, pp. 171–234, 2010. 6

[29] J. Pearl, "Probabilistic reasoning in intelligent systems," *San Mateo, CA: Kaufmann*, vol. 23, pp. 33–34, 1988. 6

[30] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Statistics and Computing*, vol. 4, no. 2, pp. 87–112, 1994. 8

[31] K. Nag and N. R. Pal, "Genetic programming for classification and feature selection," in *Evolutionary and Swarm Intelligence Algorithms*. Springer, 2019, pp. 119–141. 8

[32] W. B. Langdon and R. Poli, *Foundations of genetic programming*. Springer, 2002. 8

[33] K. Neshatian, M. Zhang, and P. Andreae, "A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming," *IEEE transactions on evolutionary computation*, vol. 16, no. 5, pp. 645–661, 2012. 8

[34] D. P. Muni, N. R. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 2, pp. 183–196, 2004. 8, 9, 18, 43

[35] P. Nordin, "A compiling genetic programming system that directly manipulates the machine code," *Advances in genetic programming*, vol. 1, pp. 311–331, 1994. 8

[36] M. O'Neill and C. Ryan, "Grammatical evolution," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 4, pp. 349–358, 2001. 8

[37] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Genetic Programming*. Springer, 2000, pp. 121–132. 8

[38] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 2, pp. 121–144, 2010. 8, 18, 43

[39] K.-H. Liu and C.-G. Xu, "A genetic programming-based approach to the classification of multiclass microarray datasets," *Bioinformatics*, vol. 25, no. 3, pp. 331–337, 2009. 9, 20, 43

[40] P. J. Rauss, J. M. Daida, and S. Chaudhary, "Classification of spectral imagery using genetic programming," in *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2000, pp. 726–733. 9, 18, 43

[41] S. A. Stanhope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," in *Evolutionary Programming VII*. Springer, 1998, pp. 735–744. 9, 18

[42] I. De Falco, A. Della Cioppa, and E. Tarantino, "Discovering interesting classification rules with genetic programming," *Applied Soft Computing*, vol. 1, no. 4, pp. 257–269, 2002. 9, 18, 43

[43] C. C. Bojarczuk, H. S. Lopes, and A. A. Freitas, "Genetic programming for knowledge discovery in chest-pain diagnosis," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 19, no. 4, pp. 38–44, 2000. 9, 18, 43

[44] H. Gray, R. Maxwell, I. Martinez-Perez, C. Arus, and S. Cerdan, "Genetic programming for classification of brain tumours from nuclear magnetic resonance biopsy spectra," *Genetic Programming*, p. 424, 1996. 9, 18

[45] D. Hope, E. Munday, and S. Smith, "Evolutionary algorithms in the classification of mammograms," in *Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on*. IEEE, 2007, pp. 258–265. 9, 18, 43

[46] G. Wilson and M. Heywood, "Introducing probabilistic adaptive mapping developmental genetic programming with redundant mappings," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 187–220, 2007. 9, 18, 43

[47] S. Sakprasat and M. C. Sinclair, "Classification rule mining for automatic credit approval using genetic programming," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 548–555. 9, 18, 43

[48] J. Kishore, L. M. Patnaik, V. Mani, and V. Agrawal, "Application of genetic programming for multicategory pattern classification," *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 3, pp. 242–258, 2000. 9, 18, 43

[49] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang, "Designing a classifier by a layered multi-population genetic programming approach," *Pattern recognition*, vol. 40, no. 8, pp. 2211–2225, 2007. 9, 18, 43

[50] M. Zhang and P. Wong, "Genetic programming for medical classification: a program simplification approach," *Genetic Programming and Evolvable Machines*, vol. 9, no. 3, pp. 229–255, 2008. 9, 18, 43

[51] M. Zhang and W. Smart, "Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification," *Pattern Recognition Letters*, vol. 27, no. 11, pp. 1266–1274, 2006. 9, 18, 43

[52] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, 2017. 9, 44

[53] I. Arnaldo, K. Krawiec, and U.-M. O'Reilly, "Multiple regression genetic programming," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 879–886. 9, 43, 44

[54] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple classifier systems*. Springer, 2000, pp. 1–15. 9, 20

[55] L. K. Hansen and P. Salamon, "Neural network ensembles," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 10, pp. 993–1001, 1990. 9, 20, 38

[56] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 3, pp. 368–386, 2013. 9, 19, 20

[57] S. Mirjalili, A. Lewis, and S. Mostaghim, "Confidence measure: a novel metric for robust meta-heuristic optimisation algorithms," *Information Sciences*, vol. 317, pp. 114–142, 2015. 9, 10, 67, 94

[58] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Transactions on evolutionary computation*, vol. 9, no. 3, pp. 303–317, 2005. 9

[59] S. Biswas, S. Das, S. Debchoudhury, and S. Kundu, "Co-evolving bee colonies by forager migration: A multi-swarm based artificial bee colony algorithm for global search space," *Applied Mathematics and Computation*, vol. 232, pp. 216–234, 2014. 9

[60] L. T. Bui, H. A. Abbass, M. Barlow, and A. Bender, "Robustness against the decision-maker's attitude to risk in problems with conflicting objectives," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 1, pp. 1–19, 2012. 10, 11, 12, 65, 66, 67, 68, 93, 94

[61] L. T. Bui, H. A. Abbass, and D. Essam, "Fitness inheritance for noisy evolutionary multi-objective optimization," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 779–785. 10

[62] C. K. Goh and K. C. Tan, "An investigation on noisy environments in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, pp. 354–381, 2007. 10, 67, 94

[63] K. Deb and H. Gupta, "Introducing robustness in multi-objective optimization," *Evolutionary Computation*, vol. 14, no. 4, pp. 463–494, 2006. 10, 66, 67, 68, 75, 76, 93, 94, 96, 104, 106, 110, 123, 124, 125, 126, 127, 129

[64] K. Deb and H. Gupta, "A constraint handling strategy for robust multi-criterion optimization," *KanGAL Report*, no. 2005001, p. 2, 2005. 10, 67, 94

[65] J. Ferreira, A. Gaspar-Cunha, C. Fonseca, and J. Covas, *Evolutionary multi-objective robust optimization*. Citeseer, 2008. 10, 67, 94

[66] A. Gaspar-Cunha and J. A. Covas, "Robustness in multi-objective optimization using evolutionary algorithms," *Computational Optimization and Applications*, vol. 39, no. 1, pp. 75–96, 2008. 10, 67, 94

[67] S. Gunawan and S. Azarm, "Multi-objective robust optimization using a sensitivity region concept," *Structural and Multidisciplinary Optimization*, vol. 29, no. 1, pp. 50–60, 2005. 10, 67, 94

[68] M. Dellnitz and K. Witting, "Computation of robust pareto points," *International Journal of Computing Science and Mathematics*, vol. 2, no. 3, pp. 243–266, 2009. 10, 67, 94

[69] Y. Xue, D. Li, W. Shan, and C. Wang, "Multi-objective robust optimization using probabilistic indices," in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, vol. 4. IEEE, 2007, pp. 466–470. 10, 67, 94

[70] L. T. Bui, D. Essam, H. A. Abbass, and D. Green, "Performance analysis of evolutionary multi-objective optimization methods in noisy environments," in *Proceedings of the 8th Asia Pacific symposium on intelligent and evolutionary systems*, 2004, pp. 29–39. 10, 67, 94

[71] E. J. Hughes, "Evolutionary multi-objective ranking with uncertainty and noise," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 329–343. 10, 67, 94

[72] J. Teich, "Pareto-front exploration with uncertain objectives," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 314–328. 10, 67, 94

[73] Y.-S. Ong, P. B. Nair, and K. Y. Lum, "Max-min surrogate-assisted evolutionary algorithm for robust design," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 392–404, 2006. 11

[74] J. Xiong, X. Tan, K.-W. Yang, and Y.-W. Chen, "Fuzzy group decision making for multiobjective problems: Tradeoff between consensus and robustness," *Journal of Applied Mathematics*, vol. 2013, 2013. 12, 66, 67, 68, 69

[75] H. Jabeen and A. R. Baig, "Review of classification using genetic programming," *International journal of engineering science and technology*, vol. 2, no. 2, pp. 94–103, 2010. 18, 43

[76] M. Muharram and G. D. Smith, "Evolutionary constructive induction," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 11, pp. 1518–1528, 2005. 18, 43

[77] K. Neshatian and M. Zhang, "Genetic programming and class-wise orthogonal transformation for dimension reduction in classification problems," in *Genetic Programming*. Springer, 2008, pp. 242–253. 18, 43

[78] H. Guo and A. K. Nandi, "Breast cancer diagnosis using genetic programming generated feature," *Pattern Recognition*, vol. 39, no. 5, pp. 980–987, 2006. 18, 43

[79] H. Guo, L. B. Jack, and A. K. Nandi, "Feature generation using genetic programming with application to fault classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 1, pp. 89–99, 2005. 18, 43

[80] K. Krawiec, "Genetic programming-based construction of features for machine learning and knowledge discovery tasks," *Genetic Programming and Evolvable Machines*, vol. 3, no. 4, pp. 329–343, 2002. 18, 43

[81] X. Tan, B. Bhanu, and Y. Lin, "Fingerprint classification based on learned features," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 35, no. 3, pp. 287–300, 2005. 18, 43

[82] M. G. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," *Genetic Programming and Evolvable Machines*, vol. 6, no. 3, pp. 265–281, 2005. 18, 43

[83] Y. Lin and B. Bhanu, "Evolutionary feature synthesis for object recognition," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 35, no. 2, pp. 156–171, 2005. 18, 43

[84] J. R. Koza, "Concept formation and decision tree induction using the genetic programming paradigm," in *Parallel Problem Solving from Nature*. Springer, 1991, pp. 124–128. 18, 43

[85] G. Folino, C. Pizzuti, and G. Spezzano, "Genetic programming and simulated annealing: A hybrid method to evolve decision trees," in *Genetic Programming*. Springer, 2000, pp. 294–303. 18, 43

[86] J. Eggermont, "Evolving fuzzy decision trees with genetic programming and clustering," in *Genetic Programming*. Springer, 2002, pp. 71–82. 18, 43

[87] H. Zhao, "A multi-objective genetic programming approach to developing pareto optimal decision trees," *Decision Support Systems*, vol. 43, no. 3, pp. 809–826, 2007. 18, 43

[88] T. M. Khoshgoftaar and Y. Liu, "A multi-objective software quality classification model using genetic programming," *Reliability, IEEE Transactions on*, vol. 56, no. 2, pp. 237–245, 2007. 18, 43

[89] A. Tsakonas, "A comparison of classification accuracy of four genetic programming-evolved intelligent structures," *Information Sciences*, vol. 176, no. 6, pp. 691–724, 2006. 18, 43

[90] C. Qing-Shan, Z. De-Fu, W. Li-Jun, and C. Huo-Wang, "A modified genetic programming for behavior scoring problem," in *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*. IEEE, 2007, pp. 535–539. 18, 43

[91] E. Carreno, G. Leguizamón, and N. Wagner, "Evolution of classification rules for comprehensible knowledge discovery," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 1261–1268. 18, 43

[92] R. R. Mendes, F. B. de Voznika, A. A. Freitas, and J. C. Nievola, "Discovering fuzzy classification rules with genetic programming and co-evolution," in *Principles of Data Mining and Knowledge Discovery*. Springer, 2001, pp. 314–325. 18, 43

[93] A. L. Garcia-Almanza and E. P. Tsang, "Evolving decision rules to predict investment opportunities," *International Journal of Automation and Computing*, vol. 5, no. 1, pp. 22–31, 2008. 19, 43

[94] U. Bhowan, M. Johnston, and M. Zhang, "Developing new fitness functions in genetic programming for classification with unbalanced data," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 406–421, 2012. 19, 20, 43

[95] W. Fu, M. Johnston, and M. Zhang, "Low-level feature extraction for edge detection using genetic programming," *IEEE transactions on cybernetics*, vol. 44, no. 8, pp. 1459–1472, 2014. 19, 43

[96] P. Wang, K. Tang, T. Weise, E. Tsang, and X. Yao, "Multiobjective genetic programming for maximizing roc performance," *Neurocomputing*, vol. 125, pp. 102–118, 2014. 19, 43, 44

[97] P. Wang, M. Emmerich, R. Li, K. Tang, T. Bäck, and X. Yao, "Convex hull-based multiobjective genetic programming for maximizing receiver operating characteristic performance," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 188–200, 2015. 19, 44

[98] P. A. Whigham and G. Dick, "Implicitly controlling bloat in genetic programming," *Evolutionary Computation, IEEE Transactions on*, vol. 14, no. 2, pp. 173–190, 2010. 19

[99] S. Luke and L. Panait, "A comparison of bloat control methods for genetic programming," *Evolutionary Computation*, vol. 14, no. 3, pp. 309–344, 2006. 19

[100] R. Poli, "A simple but theoretically-motivated method to control bloat in genetic programming," in *Genetic Programming*. Springer, 2003, pp. 204–217. 19

[101] M. J. Streeter, "The root causes of code growth in genetic programming," in *Genetic programming*. Springer, 2003, pp. 443–454. 19

[102] S. Luke and L. Panait, "Fighting bloat with nonparametric parsimony pressure," in *Parallel Problem Solving from Nature—PPSN VII*. Springer, 2002, pp. 411–421. 19

[103] F. Fernández-de Vega, G. G. Gil, J. A. G. Pulido, and J. L. Guisado, "Control of bloat in genetic programming by means of the island model," in *Parallel Problem Solving from Nature-PPSN VIII*. Springer, 2004, pp. 263–271. 19

[104] F. Fernandez, G. Galeano, and J. Gomez, "Comparing synchronous and asynchronous parallel and distributed genetic programming models," in *Genetic Programming*. Springer, 2002, pp. 326–335. 19

[105] M. Castelli, L. Vanneschi, and S. Silva, "Semantic search-based genetic programming and the effect of intron deletion," *Cybernetics, IEEE Transactions on*, vol. 44, no. 1, pp. 103–113, 2014. 19

[106] D. Rochat, M. Tomassini, and L. Vanneschi, "Dynamic size populations in distributed genetic programming," in *Genetic Programming*. Springer, 2005, pp. 50–61. 19

[107] F. Fernandez, L. Vanneschi, and M. Tomassini, "The effect of plagues in genetic programming: A study of variable-size populations," in *Genetic Programming*. Springer, 2003, pp. 317–326. 19

[108] A. Song, Q. Shi, and W. Yin, "Understanding of gp-evolved motion detectors," *Computational Intelligence Magazine, IEEE*, vol. 8, no. 1, pp. 46–55, 2013. 19

[109] J. Luna, J. Romero, C. Romero, and S. Ventura, "On the use of genetic programming for mining comprehensible rules in subgroup discovery," *Cybernetics, IEEE Transactions on*, vol. 44, no. 12, pp. 2329–2341, 2014. 19

[110] S. Pang, D. Kim, and S. Y. Bang, "Membership authentication in the dynamic group by face classification using svm ensemble," *Pattern Recognition Letters*, vol. 24, no. 1, pp. 215–225, 2003. 20

[111] H. Ishibuchi and T. Yamamoto, "Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers," in *Genetic and Evolutionary Computation-GECCO 2003*. Springer, 2003, pp. 1077–1088. 20

[112] P. Yang, Y. Hwa Yang, B. B Zhou, and A. Y Zomaya, "A review of ensemble methods in bioinformatics," *Current Bioinformatics*, vol. 5, no. 4, pp. 296–308, 2010. 20

[113] K. Nag, T. Pal, and N. Pal, "ASMiGA: An archive-based steady-state micro genetic algorithm," *Cybernetics, IEEE Transactions on*, vol. 45, no. 1, pp. 40–52, 2015. 21, 24, 26, 49, 67, 76, 96

[114] R. ENACHE, "Steady state evolutionary algorithms," Honda Research Institute Europe GmbH, Tech. Rep. HRI-EU Report 04-02, 2004. 22

[115] J.-H. Hong and S.-B. Cho, "Gene boosting for cancer classification based on gene expression profiles," *Pattern Recognition*, vol. 42, no. 9, pp. 1761–1767, 2009. 22

[116] K. Nag and T. Pal, "A new archive based steady state genetic algorithm," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–7. 24, 26, 49, 67, 76, 96

[117] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002. 24, 49, 76

[118] S. Kernell, "Presidential popularity and negative voting: An alternative explanation of the midterm congressional decline of the president's party," *The American Political Science Review*, vol. 71, no. 1, pp. 44–66, 1977. 27

[119] M. Fang, H. Takauj, S. Kaneko, and H. Watanabe, "Robust optical flow estimation for underwater image," in *Optomechatronic Technologies, 2009. ISOT 2009. International Symposium on*. IEEE, 2009, pp. 185–190. 27

[120] M. Fang, H. Takauji, and S. Kaneko, "Rapid computation of robust optical flow by efficient complementary voting," in *World Automation Congress (WAC), 2010*. IEEE, 2010, pp. 1–6. 27

[121] M. P. Fiorina and K. A. Shepsle, "Is negative voting an artifact?" *American Journal of Political Science*, pp. 423–439, 1989. 27

[122] J. Durillo, A. Nebro, and E. Alba, "The jmetal framework for multi-objective optimization: Design and architecture," in *CEC 2010*, Barcelona, Spain, 2010, pp. 4138–4325. 28, 54

[123] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997811001219 28, 54

[124] Q. Song, J. Ni, and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 1, pp. 1–14, 2013. 30, 31, 32, 35, 38, 57, 59, 60, 61, 62

[125] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proc. 20th Int'l Conf. Machine Leaning*, vol. 20, no. 2, 2003, pp. 856–863. 31, 59

[126] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *The Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004. 31, 59

[127] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999. 31, 59

[128] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003. 31, 59

[129] M. Dash, H. Liu, and H. Motoda, "Consistency based feature selection," in *Knowledge Discovery and Data Mining. Current Issues and New Applications*. Springer, 2000, pp. 98–109. 31, 59

[130] H. Almuallim and T. G. Dietterich, "Learning boolean concepts in the presence of many irrelevant features," *Artificial Intelligence*, vol. 69, no. 1, pp. 279–305, 1994. 31, 59

[131] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937. 38, 61

[132] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940. 38, 61

[133] V. Ingalalli, S. Silva, M. Castelli, and L. Vanneschi, "A multi-dimensional genetic programming approach for multi-class classification problems," in *European Conference on Genetic Programming*. Springer, 2014, pp. 48–60. 43

[134] L. Munoz, S. Silva, and L. Trujillo, "M3gp–multiclass classification with gp," in *European Conference on Genetic Programming*. Springer, 2015, pp. 78–91. 43

[135] H. Al-Sahaf, M. Zhang, and M. Johnston, "Binary image classification: A genetic programming approach to the problem of limited training instances," *Evolutionary computation*, vol. 24, no. 1, pp. 143–182, 2016. 43

[136] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016. 43

[137] L. Liao, "Discovering prognostic features using genetic programming in remaining useful life prediction," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 5, pp. 2464–2472, 2014. 43

[138] J. G. Moreno-Torres, X. Llorà, D. E. Goldberg, and R. Bhargava, "Repairing fractures between data using genetic programming-based feature extraction: A case study in cancer diagnosis," *Information Sciences*, vol. 222, pp. 805–823, 2013. 43

[139] C. Estébanez, J. M. Valls, and R. Aler, "Gppe: a method to generate ad-hoc feature extractors for prediction in financial domains," *Applied Intelligence*, vol. 29, no. 2, pp. 174–185, 2008. 43

[140] Y. Zhang and P. I. Rockett, "Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 795–802. 43

[141] L. Guo, D. Rivero, J. Dorado, C. R. Munteanu, and A. Pazos, "Automatic feature extraction using genetic programming: an application to epileptic eeg classification," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 425–10 436, 2011. 43

[142] B. Tran, B. Xue, and M. Zhang, "Genetic programming for feature construction and selection in classification on high-dimensional data," *Memetic Computing*, vol. 8, no. 1, pp. 3–15, 2016. 43

[143] W. Fu, M. Johnston, and M. Zhang, "Distribution-based invariant feature construction using genetic programming for edge detection," *Soft computing*, vol. 19, no. 8, pp. 2371–2389, 2015. 43

[144] Y. Liang, M. Zhang, and W. N. Browne, "Image feature selection using genetic programming for figure-ground segmentation," *Engineering Applications of Artificial Intelligence*, vol. 62, pp. 96–108, 2017. 43

[145] A. Cano, S. Ventura, and K. J. Cios, "Multi-objective genetic programming for feature extraction and data visualization," *Soft Computing*, vol. 21, no. 8, pp. 2069–2089, 2017. 44

[146] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004. 44

[147] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri *et al.*, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *science*, vol. 286, no. 5439, pp. 531–537, 1999. 47

[148] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008. 54, 55

[149] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001. 59

[150] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *Icml*, vol. 96. Citeseer, 1996, pp. 148–156. 59

[151] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 59

[152] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011. 62

[153] L. Zhihuan, L. Yinhong, and D. Xianzhong, "Non-dominated sorting genetic algorithm-ii for robust multi-objective optimal reactive power dispatch," *IET Generation, Transmission Distribution*, vol. 4, no. 9, pp. 1000–1008, 2010. 66

[154] K. Deb *et al.*, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons Chichester, 2001, vol. 2012. 67

[155] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen *et al.*, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007, vol. 5. 67

[156] U. Bose, A. M. Davey, and D. L. Olson, "Multi-attribute utility methods in group decision making: past applications and potential for inclusion in gdss," *Omega*, vol. 25, no. 6, pp. 691–706, 1997. 67

[157] J. Lu and D. Ruan, *Multi-objective group decision making: methods, software and applications with fuzzy set techniques*. Imperial College Press, 2007, vol. 6. 67

[158] G. Zhang, J. Ma, and J. Lu, "Emergency management evaluation by a fuzzy multi-criteria group decision support system," *Stochastic Environmental Research and Risk Assessment*, vol. 23, no. 4, pp. 517–527, 2009. 67

[159] F. Chiclana, J. T. GarcíA, M. J. del Moral, and E. Herrera-Viedma, "A statistical comparative study of different similarity measures of consensus in group decision making," *Information Sciences*, vol. 221, pp. 110–123, 2013. 67

[160] F. J. Cabrerizo, R. Ureña, W. Pedrycz, and E. Herrera-Viedma, "Building consensus in group decision making with an allocation of information granularity," *Fuzzy Sets and Systems*, vol. 255, pp. 115–127, 2014. 67

[161] E. Herrera-Viedma, F. J. Cabrerizo, J. Kacprzyk, and W. Pedrycz, "A review of soft consensus models in a fuzzy environment," *Information Fusion*, vol. 17, pp. 4–13, 2014. 67

[162] E. Herrera-Viedma, F. Herrera, and F. Chiclana, "A consensus model for multiperson decision making with different preference structures," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 32, no. 3, pp. 394–402, 2002. 67

[163] I. J. Pérez, F. J. Cabrerizo, S. Alonso, and E. Herrera-Viedma, "A new consensus model for group decision making problems with non-homogeneous experts," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 4, pp. 494–498, 2014. 67

[164] E. Herrera-Viedma, S. Alonso, F. Chiclana, and F. Herrera, "A consensus model for group decision making with incomplete fuzzy preference relations," *IEEE Transactions on fuzzy Systems*, vol. 15, no. 5, pp. 863–877, 2007. 67

[165] D. Dubois and H. Prade, "Weighted minimum and maximum operations in fuzzy set theory," *Information Sciences*, vol. 39, no. 2, pp. 205–210, 1986. 69

[166] R. R. Yager, "Weighted triangular norms using generating functions," *International Journal of Intelligent Systems*, vol. 19, no. 3, pp. 217–231, 2004. 69

[167] K. Deb and S. Tiwari, "Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1062–1087, 2008. 76

[168] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009. 76

[169] M. Iwatsuki, M. Kawamata, and T. Higuchi, "Statistical sensitivity and minimum sensitivity structures with fewer coefficients in discrete time linear systems," *IEEE transactions on circuits and systems*, vol. 37, no. 1, pp. 72–80, 1990. 93, 100

[170] J. Y. Choi and C.-H. Choi, "Sensitivity analysis of multilayer perceptron with differentiable activation functions," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 101–107, 1992. 93, 100

[171] C. Barrico and C. Antunes, "A new approach to robustness analysis in multi-objective optimization," in *7th International Conference on Multi-Objective Programming and Goal Programming (MOPGP 2006), Loire Valley, City of Tours, France, June*, 2006, pp. 12–14. 94

[172] C. Barrico and C. H. Antunes, "Robustness analysis in multi-objective optimization using a degree of robustness concept," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 2006, pp. 1887–1892. 94

[173] Y. Jin and B. Sendhoff, "Trade-off between performance and robustness: an evolutionary multiobjective approach," in *international conference on Evolutionary Multi-Criterion Optimization*. Springer, 2003, pp. 237–251. 94, 95, 100, 115

[174] T. Ray, "Constrained robust optimal design using a multiobjective evolutionary algorithm," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 1. IEEE, 2002, pp. 419–424. 94