Multiobjective Optimization — New Formulation and Application to Radar Signal Processing

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

Telematics and Signal Processing

By

VIKAS BAGHEL

Roll No: 207EC105

Department of Electronics and Communication

National Institute of Technology

Rourkela, India

2009

Multiobjective Optimization — New Formulation and Application to Radar Signal Processing

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

Telematics and Signal Processing

By

VIKAS BAGHEL

Roll No: 207EC105

Under the Supervision of

Prof. Ganapati Panda FNAE, FNASc.

Department of Electronics and Communication

National Institute of Technology

Rourkela, India

2009

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

CERTIFICATE

This is to certify that the thesis entitled, **"Multiobjective Optimization — New Formulation and Application to Radar Signal Processing "** submitted by **Vikas Baghel** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electronics & Communication Engineering** with specialization in **Telematics and Signal Processing** during 2008-2009 at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date

Prof. G. Panda (FNAE, FNASc)

Dept. of Electronics & Communication Engg.

National Institute of Technology

Rourkela-769008

Orissa, India

Acknowledgment

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Ganapati Panda, who is supported me throughout my project with patience and knowledge whilst allowing me the room to work in my own way. Besides our weekly meetings, he was always willing to answer my questions and to help me to overcome my doubts. I feel proud that I am able to finish his toughest classes successfully. He always pushed me to my limits to get the best out of me. I attribute the level of my Masters degree to his encouragement and efforts and without him this thesis, too, would not have been written or completed. One simply can not wish for a better or friendlier supervisor.

Sincere thanks to Prof. S. K. Patra, Prof. K. K. Mahapatra, Prof. G. S. Rath, Prof. S. Meher , Prof. S.K.Behera, Prof. Punam Singh and Prof. Ajit Kumar Sahoo for their constant cooperation and encouragement through out the course.

I am privileged to express my sincere gratitude to Prof. K. Rajarajeswari for their inspiration and co-operation in collecting secondary information. I gratefully acknowledge Mr. P. Srihari for his advice, supervision, and crucial contribution, which made him a backbone of this research and so to this thesis.

I also extend my thanks to entire faculty of Department of Electronics and Communication Engineering, National Institute of Technology Rourkela, Rourkela who have encouraged me throughout the course of Masters Degree.

I would like to thank my friends, especially Satyasai Jagannath Nanda, Pyari Mohan Pradhan, Nithin V george, Pavan kumar Gorpuni, S.K.Verma and Jyoti for their help during the course of this work. I am also thankful to all my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious.

And finally thanks to my parents and my sisters, whose faith, patience and teaching had always inspired me to walk upright in my life. Without all these beautiful people my world would have been an empty place.

Vikas Baghel

CONTENTS

Abstract

The present thesis aims to make an in-depth study of Multiobjective optimization (MOO), Multiobjective algorithms and Radar Pulse Compression. Following the approach of bacteria foraging technique, a new MOO algorithm Multiobjective Bacteria Foraging Optimization (MOBFO) has been proposed in this thesis. We compared the performance of our proposed algorithm with existing algorithms Nondominated Sorting Genetic Algorithm (NSGA-II) and Multiobjective Particle Swarm Optimization (MOPSO) for different test functions. In radar signal processing Pulse Compression is used for high range resolution and long range detection. The classical methods for Pulse Compression of the received signal use matched filter and mismatched filter. For improving the performance of pulse compression, a new problem formulation has been constructed that uses constrained function optimization with the help of Particle Swarm Optimization (PSO). Artificial Neural Network (ANN) is being used for Pulse Compression that achieves a significant supression of the sidelobes. Functional Link Artificial Neural Network (FLANN) has been proposed for better sidelobes reduction than Multi Layer Perceptron (MLP) network with both lower computational and lower structural complexity. MOO approach has been proposed to use with Radial Basis Function (RBF) for Pulse Compression that improves the accuracy and complexity of RBF network.

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

CHAPTER 1

INTRODUCTION

1.1 Background

Most real-world search and optimization problems are naturally posed as multiobjective optimization problem (MOP). The majority of engineering optimization is the MOP, sometimes it need to make multiple targets all reach the optimal in a given region, but it is regrettable that goals are generally conflicting. A MOP differs from a single-objective optimization problem because it contains several objectives that require optimization. When optimizing a single objective problem, the best single design solution is the goal. But for MOP, with several (possibly conflicting) objectives, there is usually no single optimal solution. Therefore, the decision maker is required to select a solution from a finite set by making compromises. A suitable solution should provide for acceptable performance over all objectives.

Many fields continue to address complex real-world MOPs using search techniques developed within computer engineering, computer science, decision sciences, and operations research. The potential of evolutionary algorithms for solving MOPs was hinted as early as the late 1960s by Rosenberg [1.1]. However, the first actual implementation of a multiobjective evolutionary algorithm (MOEA) was produced until the mid-1980s by David Schaffer in his doctoral dissertation [1.2]. Since then, a considerable amount of research has been done in this area, now known as evolutionary multiobjective optimization (EMOO).

To improve the performance in pulse radar detection, pulse compression technique [1.3], which involves the transmission of a long duration wide bandwidth signal code, and the compression of this received signal to a narrow pulse, is always employed. In practice two different approaches are used to obtain pulse compression. The first one is to use a matched filter; here codes with small sidelobes in their autocorrelation function (ACF) are used [1.3, 1.4]. The second approach to pulse compression is to use inverse filters of two kinds, namely, non-recursive time invariant causal filter [1.5] and recursive time variant filter [1.6]. Two different approaches using a multi-layered neural network, which yield better signal-to-sidelobe ratio (SSR) (the ratio of peak signal to maximum sidelobe) than the traditional approaches have been reported in [1.7-1.9]. In the first, a multi-layered neural network approach using back-propagation (BP) as the learning algorithm is used [1.7, 1.10]. Whereas in the second approach, Radial Basis Function (RBF) network approach, using supervised selection of centers learning strategies, has been used [1.11].

1.2 Motivation

The main motivation for using MOEAs to solve MOPs is because MOEAs deal simultaneously with a set of possible solutions (the so-called population) which allows us to find several members of the Pareto optimal set in a single run of the algorithm, instead of having to perform a series of separate runs as in the case of the traditional mathematical programming techniques [1.12, 1.13]. The Pareto-optimal front can be exploited to select solutions appropriate for each particular application without having to weight the objectives in advance or reduce the multiple objectives in some other way. Additionally, MOEAs are less susceptible to the shape or continuity of the Pareto front (e.g. , they can easily deal with discontinuous and concave Pareto fronts), whereas these two issues are known problems with mathematical programming.

Artificial Neural Networks (ANNs) are computational tools inspired by biological nervous system with applications in science and engineering[1.14 - 1.17]. This work is about a kind of ANN for applications in multivariate nonlinear regression, classification and times-series called Radial Basis Function (RBF) Network [1.14, 1.18]. Although ANNs have usually achieved good performances in several domains, those performances and the ANN training process are directly influenced by an appropriate choice of the network architecture. Unfortunately, the space of network architectures is infinite and complicated and there is no general purpose, reliable, and automatic method to search that large space. When the Trial and Error method [1.19, 1.20] is employed, different values for the network parameters must be selected, trained and compared before the choice of an ultimate network. The disadvantage of this method becomes more apparent if, after the choice of the best values, the patterns set is changed, making necessary to restart the design process. This search can be done more efficient if heuristics are used to guide it. Multiobjective optimization approach is one of the standard techniques of searching in complicated search spaces [1.21]. So one of the reasons to apply MOGAs to the RBF Networks is due to the complex nature of their optimization which involves aspects of both numerical and combinatorial optimization with two objective functions, one for accuracy and other for complexity of network [1.22].

1.3 Present Work

This thesis is organized in such a way that its contents provides a general overview of the field now called evolutionary multiobjective optimization (EMO), which refers to the use of evolutionary algorithms to solve multiobjective optimization problems. We have proposed a new MOO algorithm Multi Objective Bacteria Foraging Optimization (MOBFO), following the approach of bacteria foraging technique. We have compared the performance of our proposed algorithm with existing algorithms NSGA-II and MOPSO for different test functions. For improving the performance of pulse compression in Radar detection, we have constructed a new problem formulation that uses constrained function optimization with the help of Particle Swarm Optimization (PSO). Traditionally, Artificial Neural Network (ANN) is being used for Pulse Compression that achieves a significant suppression of the sidelobes. Functional Link Artificial Neural Network (FLANN) has been proposed by us for better sidelobes reduction than Multi Layer Perceptron (MLP) network with both lower computational and lower structural complexity. We have also proposed the use of MOO approach to use with Radial Basis Function (RBF) for Pulse Compression that improves the accuracy and structure complexity of RBF.

1.4 Thesis Organization

Chapter-1 Introduction

Chapter-2 Multiobjective Optimization

Most multi-objective optimization methods use the concept of domination in their search. Thus, in this chapter, we have presented definitions for domination, a non-dominated set and a Pareto-optimal set of solutions.

Chapter-3 Multiobjective Evolutionary Algorithms

In this chapter, we have reviewed two most popular multiobjective optimization algorithms, NSGA-II and MOPSO. Here, we also proposed a new multiobjective algorithms based on the bacteria foraging technique, named Multiobjective Bacteria Foraging Optimization (MOBFO).

Chapter-4 Performance Evaluation of Multi-objective Evolutionary Algorithms

In this chapter we evaluated the performance of multiobjective algorithms (NSGA II, MOPSO, and proposed MOBFO) based on computational time, convergence metric and diversity metric for the different standard test functions.

Chapter-5 Application : Radar Pulse Compression

This chapter presents the basic concept of the pulse compression for the long range detection as well as for high range resolution in the radar detection technique. Our work on Pulse Compression using PSO and FLANN has been discussed. Then we compared our proposed methods with classical methods and with ANN subsequently in this chapter.

Chapter-6 Pulse Compression using RBF Network : A Multiobjective approach

This chapter deals with the multiobjective concept using structure selection of RBF network used for Pulse Compression of transmitted coded signals with different lengths. We have discussed the result of comparison of this scheme with fixed centered RBF Network.

Chapter-7 Conclusion and Scope for Future Work

The overall conclusion of the thesis is presented in this chapter. It also contains some future research topics which need attention and further investigation.

Chapter 2

Multiobjective Optimization

CHAPTER 2

MULTIOBJECTIVE OPTIMIZATION

2.1 Introduction

Most real-world engineering optimization problems are multiobjective in nature, since they normally have several (possibly conflicting) objectives that must be satisfied at the same time. The notion of 'optimum' has to be re-defined in this context and instead of aiming to find a single solution, we will try to produce a set of good compromises or trade-offs from which the decision maker will select one.

Due to increasing interest in solving real-world multiobjective optimization problems using evolutionary algorithms (EA), researchers have developed a number of evolutionary multiobjective algorithms (EMO) based on real parameters. The presence of multiple objectives in a problem, in principle, gives rise to a set of optimal solutions (largely known as Pareto-optimal solutions), instead of a single optimal solution. In the absence of any further information, one of these Pareto-optimal solutions cannot be said to be better than the other. This demands a user to find as many Pareto-optimal solutions as possible. Classical optimization methods (including the multi-criterion decision-making methods) suggest converting the multiobjective optimization problem to a single objective optimization problem by emphasizing one particular Pareto-optimal solution at a time. When such a method is to be used for finding multiple solutions, it has to be applied many times, hopefully finding a different solution at each simulation run. Over the past decade, a number of multiobjective evolutionary algorithms (MOEAs) have been suggested [2.1 - 2.4]. The primary reason for this is their ability to find multiple Pareto-optimal solutions in one single simulation run. Since evolutionary algorithms (EAs) work with a population of solutions, a simple EA can be extended to maintain a diverse set of solutions. With an emphasis for moving toward the true Pareto-optimal region, an EA can be used to find multiple Pareto-optimal solutions in one single simulation run.

2.2 Definitions

In order to develop an understanding of MOPs and the ability to design MOEAs to solve them, a series of formal non-ambiguous definitions are require. These definitions provide a precise set of symbols and formal relationship that permit proper analysis of MOEA structures and associated testing and evaluation. Moreover, they are related to primary goals for a MOEA [2.5]:

- Preserve nondominated points in objective space and associated solution points in decision space.
- Continue to make algorithmic progress towards the Pareto Front in objective function space.
- Maintain diversity of points on Pareto front (phenotype space) and/or of Pareto optimal solutions - decision space (genotype space).
- Provide the decision maker (DM) “enough” but limited number of Pareto points for selection resulting in decision variable values.

2.2.1 Single Objective Optimization

General Single Objective Optimization Problem

Definition 1 (General Single Objective Optimization Problem): When an optimization problem modeling a physical system involves only one objective function, the task of finding the optimal solution is called “ *single objective optimization*”.

A general single objective optimization problem is defined as minimizing (or maximizing) $f(\mathbf{x})$ subject to $g_i(\mathbf{x}) \leq 0, i = \{1, \ldots, m\}$, and $h_j(\mathbf{x}) = 0, j = \{1, \ldots, p\}, \mathbf{x} \in \Omega$. A solution minimizes (or maximizes) the scalar $f(\mathbf{x})$ where $\mathbf{x}$ is a n-dimensional decision variable vector $\mathbf{x} = (x_1, \ldots, x_n)$ from some universe Ω.

Observe that $g_i(\mathbf{x}) \leq 0$ and $h_j(\mathbf{x}) = 0$ represent constraints that must be fulfilled while optimizing (minimizing or maximizing) $f(\mathbf{x})$. Ω contains all possible $\mathbf{x}$ that can be used to satisfy an evaluation of $f(\mathbf{x})$ and its constraints. Of course, $\mathbf{x}$ can be a vector of continuous or discrete variables as well as f being continuous or discrete.

Definition 2 (Single Objective Global Minimum Optimization): Given a function $f : \Omega \subseteq \Re^n \rightarrow \Re, \Omega \neq \phi$, for $\mathbf{x} \in \Omega$ the value $f^* \triangleq f(\mathbf{x}^*) > -\infty$ is called a **global minimum** *if and only if*

$$\forall\ \mathbf{x} \in \Omega : f(\mathbf{x}^*) \leq f(\mathbf{x}) \tag{2.1}$$

$\mathbf{x}^*$ is by definition the global minimum solution, f is the objective function, and the set Ω is the feasible region of $\mathbf{x}$. The goal of the determining the global minimum solution(s) is called the **global optimization problem** for a single objective problem.

Although single objective optimization problems may have a unique optimal solution, MOPs (as a rule) present a possible uncountable set of solutions, which when evaluated, produce vectors whose components represent trade-offs in objective space.

2.2.2 Multiobjective Optimization

The **Multiobjective Optimization Problem** (also called multi-criteria optimization, multi-performance or vector optimization problem) can then be defined (in words) as the problem of finding [2.6]:

"A vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions from a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the decision maker."

The mathematical definition of a MOP is important in providing a foundation of understanding between the interdisciplinary nature of deriving possible solution techniques (deterministic, stochastic); i.e.,search algorithms. The following discussions present generic MOP mathematical and formal symbolic definitions.

The single objective formulation is extended to reflect the nature of multiobjective problems where there is not one objective function to optimize, but many. Thus, there is not one unique solution but set of solutions. This set of solutions are found through the use of Pareto Optimality Theory [2.7]. Note that multiobjective problems require a decision marker to make a choice of $\mathbf{x}_i^*$ values. The selection is essentially a trade-off of one complete solution $\mathbf{x}$ over another in multiobjective space.

More precisely, MOPs are those problems where the goal is to optimize k objective functions simultaneously. This may involve the maximization of all k functions, the minimization of all k functions or a combination of maximization and minimization of these k functions. A MOP global minimum (or maximum) problem is formally defined in definition 3 [2.5-2.8].

Definition 3 (General MOP): A **general MOP** is defined as

$$\begin{aligned} Minimizing(or Maximizing)\ & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \\ subject\ to\ & g_i(\mathbf{x}) \leq 0,\ i = \{1, \ldots, m\} \\ and\ & h_j(\mathbf{x}) = 0,\ j = \{1, \ldots, p\} \end{aligned} \tag{2.2}$$

A MOP solutions minimizes (or maximizes) the components of a vector $\mathbf{F}(\mathbf{x})$ where $\mathbf{x}$ is a n-dimensional decision variable vector $\mathbf{x} = (x_1, \ldots, x_n)$ from some universe Ω. It is noted that $g_i(\mathbf{x}) \leq 0$, and $h_j(\mathbf{x}) = 0$ represent constraints that must be full filled while minimizing (or maximizing) $\mathbf{F}(\mathbf{x})$ and Ω contains all possible $\mathbf{x}$ that can be used to satisfy an evaluation of $\mathbf{F}(\mathbf{x})$.

Thus, a MOP consists of k objectives reflected in the k objective functions, $m + p$ constraints on the objective functions and n decision variables. The k objective functions may be linear or nonlinear and continuous or discrete in nature. The evaluation function, $\mathbf{F} : \Omega \rightarrow \wedge$, is a mapping from the vector of decision variables ($\mathbf{x} = x_1, \ldots, x_n$) to output vectors ($\mathbf{y} = a_1, \ldots, a_k$). Of course, the vector of decision variables $\mathbf{x}_i$ can be continuous or discrete.

Pareto Terminology

Having several objective functions, the notation of "optimum" changes, because in MOPs, the aim is to find compromises (or "tradeoff") rather than a single solution as in global optimization. The notion of "optimum" most commonly adopted is that originally proposed by Francis Ysidro Edgeworth and later generalized by Vilfredo Pareto.

Definition 4 (Pareto Optimality): A solution $\mathbf{x} \in \Omega$ is said to be Pareto optimal with respect to (w.r.t.) Ω if and only if there is no $\mathbf{x}' \in \Omega$ for which $\mathbf{v} = \mathbf{F}(\mathbf{x}') = (f_1(\mathbf{x}'), \ldots, f_k(\mathbf{x}'))$ dominates $\mathbf{u} = \mathbf{F(x)} = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$. The phrase **Pareto optimal** is taken to mean with respect to the entire decision variable space unless otherwise specified.

In words, this definition says that $\mathbf{x}^*$ is Pareto optimal if there exists no feasible vector $\mathbf{x}$ which would decrease some criterion without causing a simultaneous increase in at least one criterion (assuming minimization).

The concept of Pareto optimality is integral to the theory and the solving of MOPs. Additionally, there are a few more definitions that are also adopted in multiobjective optimization [2.4-2.7]:

Definition 5 (Pareto Dominance): A vector $\mathbf{u} = (u_1, \ldots, u_k)$ is said to dominate another vector $\mathbf{v} = (v_1, \ldots, v_k)$ (denoted by $\mathbf{u} \preceq \mathbf{v}$) if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e.,

$$\begin{aligned} &\forall i \in \{1, \ldots, k\}, u_i \leq v_i \\ &\wedge \exists i \in \{1, \ldots, k\} : u_i < v_i \end{aligned} \tag{2.3}$$

Definition 6 (Pareto Optimal set): For a given MOP, $\mathbf{F(x)}$, **the Pareto Optimal Set**, $\mathbf{P}^*$, is defined as:

$$\mathbf{P}^* := \{\mathbf{x} \in \Omega | \neg\exists\ \mathbf{x}' \in \Omega\ \mathbf{F}(\mathbf{x}') \leq \mathbf{F}(\mathbf{x})\} \tag{2.4}$$

Pareto optimal solutions are those solutions within the genotype search space (decision space) whose corresponding phenotype objective vector components cannot be all simultaneously improved. These solutions are also termed non-inferior, admissible, or efficient solutions, with the entire set represented by P^*. Their corresponding vectors are termed *nondominated*; selecting a vector(s) from this vector set (the Pareto front set PF^* implicitly indicates acceptable Pareto optimal solutions, decision variables or genotypes. These solutions may have no apparent relationship besides their membership in the Pareto optimal set. They form the set of all solutions whose associated vectors are nondominated; Pareto optimal solutions are classified as such based on their evaluated functional values.

Definition 7 (Pareto Front): For a given MOP, $\mathbf{F(x)}$, and Pareto Optimal set, $\mathbf{P}^*$, the **Pareto Front** $\mathbf{PF}^*$ defined as:

$$\mathbf{PF}^* := \{\mathbf{u} = \mathbf{F}(\mathbf{x}) | \mathbf{x} \in \mathbf{P}^*\} \tag{2.5}$$

When plotted in objective space, the nondominated vectors are collectively known as the **Pareto Front**. Again, P^*, is a subset of some solution set. Its evaluated objective vectors form PF^*, of which each is nondominated with respect to all objective vectors produced by evaluating every possible solution in Ω. In general, it is not easy to find an analytical expression of the line or surface that contains these points and in most cases, it turns out to be impossible. The normal procedure to generate the Pareto front is to compute many points in Ω and their corresponding $f(\Omega)$. When there is a sufficient number of these, it is then possible to determine the nondominated points and to produce

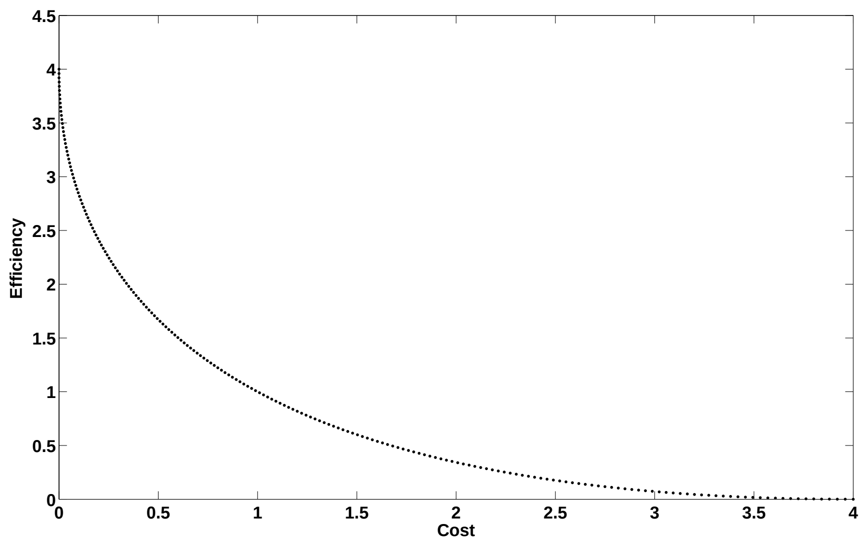


Figure 2.1: An example of a MOP with two objective functions: Cost and Efficiency

.

the Pareto front. A sample Pareto front is shown in the Fig. ??. The Pareto front or trade-off surface is delineated by a curved surface.

Although single objective optimization problems may have a unique optimal solution, MOPs usually have a possibly uncountable set of solutions on a Pareto front. Each solution associated with a point on than Pareto front is a vector whose components represent trade-offs in the decision space or Pareto solution space.

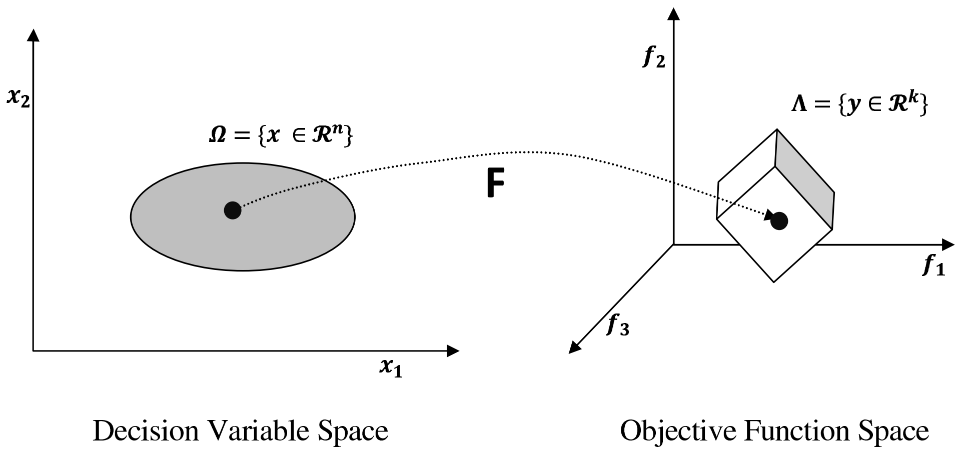


Figure 2.2: MOP Evaluation Mapping

The MOPs evaluation function, $f : \Omega \rightarrow \wedge$, maps decision variables ($\mathbf{x} = x_1, \ldots, x_n$) to the vectors ($\mathbf{y} = a_1, \ldots, a_k$). This situation is represented in Fig. ?? for the case $n = 2, m = 0$, and $k = 3$. This mapping may or may not be onto some region of objective function space, dependents upon the functions and constraints composing the particular MOP.

Note that the DM is often selecting solution via choice of acceptable objective performance, represented by the Pareto front. Choosing an MOP solution that optimizes only one objective may well ignore solutions, which from an overall standpoint, are "better". The Pareto optimal set contains those better solutions. Identifying a set of Pareto optimal solutions is thus key for a DM's selection of a "compromise" solution satisfying the objectives as "best" possible. Of course, the accuracy of the decision marker's view depends on both the true Pareto optimal set and the set presented as Pareto optimal.

2.3 Summary

This chapter contains the basic definitions and formal notation that are adopted throughout this thesis. Formal definitions of the general multi-objective optimization and the concept of Pareto optimum are provided. Other related concepts such as Pareto optimal set, Pareto front are also introduced.

Chapter 3

Multiobjective Evolutionary Algorithms

CHAPTER 3

MULTIOBJECTIVE EVOLUTIONARY ALGORITHMS

3.1 Introduction

Evolutionary algorithms (EA's) mimic natural evolutionary principles to constitute search and optimization procedures. EA's are often well-suited for optimization problems involving several, often conflicting objectives. Real world engineering optimization problems involve multiple design factors and constraints and consist in minimizing multiple non-commensurable and often competing objectives. In recent years, many evolutionary techniques for multiobjective optimization have been proposed [3.1]. In this chapter, we will present two popular Multiobjective evolutionary algorithms, Nondominated Sorting Genetic Algorithm-II (NSGA-II) [3.4] and Multiobjective Particle Swarm Optimization (MOPSO) [3.7], and one proposed "Multiobjective Bacteria Foraging Optimization (MOBFO)" algorithm.

3.2 Nondominated Sorting Genetic Algorithm-II

Genetic algorithms (GA's) are search and optimization procedures that motivated by the principles of natural genetics and natural selection [3.3]. In this section multiobjective optimization problems solved by an evolutionary algorithm, called Nondominated Sorting Genetic Algorithm [3.2, 3.4]. It is a fast nondominated sorting approach with low computational complexity. It's come in to category of Elitist multiobjective evolutionary algorithm [3.6]. As the name suggest, an elite-preserving operator favors the elites of a population by giving them an opportunity to be directly carried over the next generation. In this way, a good solution found early on in the run will never be lost unless a better solution is discovered.

3.2.1 Fast Nondominated Sorting Approach

The dual objectives in a multiobjective optimization algorithm are maintained by using a fitness assignment scheme which prefers non-dominated solutions. In the fast nondomination approach, the population is sorted based on the nondomination. Each solution is assigned a fitness (or rank) equal to its nondomination level (1 is the best level, 2 is the next-best level, and so on). For the population size of N and for M objective functions, in the following paragraph we describe a fast non-dominated sorting approach [3.4].

First, for each solution we calculate two entities:

1) *Domination count*, n_p the number of solutions which dominate the solution, and

2) S_p, a set of solutions which the solution p dominates.

An Nondominated Sorting Approach:

Step 1 For each solution $p \in P$, $n_p = 0$ and initialize $S_p = \Phi$. For all $q \neq p$ and $q \in P$, perform Step 2 and then proceed to step 3.

Step 2 If $p \preceq q$, update $S_p = S_p \cup q$. Otherwise, if $q \preceq p$, set $n_p = n_p + 1$.

Step 3 If $n_p = 0$, keep p in the first non-dominated front F_1. Set a front counter $i = 1$.

Step 4 While $F_i \neq \Phi$, perform the following steps.

Step 5 Initialize $Q = \Phi$ for storing next non-dominated solutions. For each $p \in P_i$ and for each $q \in S_p$,

  Step 5a Update $n_q = n_q - 1$.

  Step 5b If $n_q = 0$, keep q in Q, or perform $Q = Q \cup \{q\}$.

Step 6 Set $i = i + 1$ and $F_i = Q$. Go to step 4.

Steps 1 to 3 find the solutions in the first non-dominated front and require $O(MN^2)$ computational complexity. Steps 4 to 6 repeatedly find higher fronts and require at most $O(N^2)$ comparisons.

3.2.2 Diversity Preservation

It is desired that an EA maintains a good spread of solutions in the obtained set of solutions. In the proposed MOGA, with a crowded-comparison approach that eliminates the above difficulty to some extent.

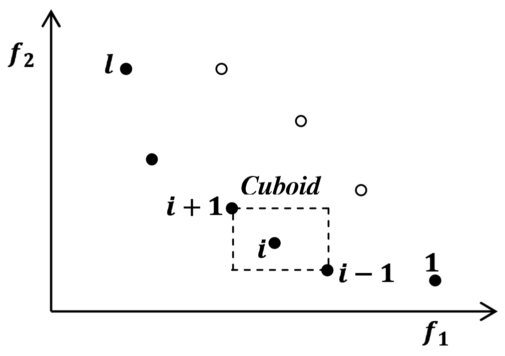


Figure 3.1: Crowding-distance Calculation

Density Estimation

To get an estimate of the density of solutions surrounding a particular solution in the population, we calculate the average distance of two points on either side of this point along each of the objectives. In Fig. ??, the crowding distance of the i^{th} solution in its front is the average side length of the cuboid (shown with a dashed box).

The algorithm as shown below outlines the crowding-distance computation procedure of all solutions in a nondominated set I.

Step 1: Call the number of solution in I as $l = |I|$. For each solution i in the set, first assign $d_i = 0$.

Step 2: For each objective function $m = 1, 2, \ldots, M$, sort the set in worse order of f_m or, find the sorted indices vector: $I^m = sort(f_m, >)$.

Step 3: For $m = 1, 2, \ldots, M$, assign a large distance to the boundary solutions, or $d_{I_1^m} = d_{I_l^m} = \infty$, and for all other solutions $j = 2$ to $(l-1)$, assign

$$d_{I_j^m} = d_{I_j^m} + (f_m^{I_{j+1}^m} - f_m^{I_{j-1}^m})/(f_m^{max} - f_m^{min}) \tag{3.1}$$

The index I_j denotes the solution index of the j-th member in the sorted list and the parameters f_m^{max} and f_m^{min} are the maximum and minimum values of the m^{th} objective function.

A solution with a smaller value of this distance measure is, in some sense, more crowded by other solutions. This is exactly what we compare in the proposed crowded-comparison operator, described below.

Crowded-Comparison Operator

The crowded-comparison operator ($\preceq_n$) guides the selection process at the various stages of the algorithm toward a uniformly spread-out Pareto optimal front. Assume that every

individual i in the population has two attributes:
1) Nondomination rank (i_{rank})
2) Crowding distance ($i_{distance}$)

We now define a partial order as:

$$(i \preceq_n j) \begin{cases} \text{if } (i_{rank} < j_{rank}) \\ \text{or } (i_{rank} = j_{rank}) \end{cases}$$

That is, between two solutions with differing nondomination ranks, we prefer the solution with the lower (better) rank. Otherwise, if both solutions belong to the same front, then we prefer the solution that is located in a lesser crowded region.

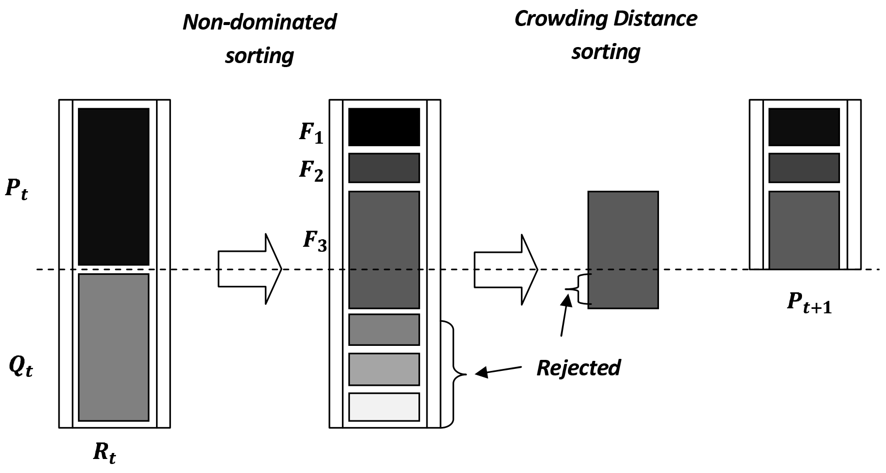


Figure 3.2: NSGA Procedure

3.2.3 Main Loop

Initially, a random parent population P_0 is created. The population is sorted in to different non-domination levels. Each solution is assigned a fitness (or rank) equal to its non-domination level (1 is the best level, 2 is the next-best level, and so on). Thus, minimization of fitness is assumed. Binary tournament selection (with a crowded tournament operator), recombination, and mutation operators [3.5] are used to create an offspring population Q_0 of size N. The NSGA-II procedure is outlined in the following (Fig. **??**):

Step 1: Combined population and offspring and create $R_t = P_t \cup Q_t$. Perform a non-dominated sorting to R_t and identify different non-dominated fronts $F_i,\ i = 1, \ldots$,etc.

Step 2: Set new population $P_{t+1} = \phi$. Set a counter i = 1. Until $|P_{t+1}| + |F_i| < N$, perform $P_{t+1} = P_{t+1} + F_i$ and $i = i + 1$.

Step 3: Perform the Crowding-sort $(F_i, \preceq_n)$ procedure and include the most widely spread $(N - |P_{t+1}|)$ solutions by using the crowding distance values in the sorted F_i to P_{t+1}.

Step 4: Create offspring population Q_{t+1} from P_{t+1} by using the crowded tournament selection, crossover and mutation operators.

3.3 Multiobjective Particle Swarm Optimization

Multiobjective Particle Swarm Optimization (MOPSO), an approach in which Pareto dominance is incorporated into particle swarm optimization (PSO) in order to allow this heuristic to handle problems with several objective functions. Here it uses a secondary (i.e., external) repository of particles that is later used by other particles to guide their own flight.

Kennedy and Eberhart [3.8] initially proposed the swarm strategy for optimization. Particle swarm optimization (PSO) is a stochastic optimization technique that draws inspiration from the behavior of a flock of birds or the collective intelligence of a group of social insects with limited individual capabilities. In PSO, individuals, referred to as particles, are "flown" through hyper dimensional search space. Changes to the position of the particles within the search space are based on the social psychological tendency of individuals to emulate the success of other individuals. A swarm consists of a set of particles, where each particle represents a potential solution. The position of each particle is changed according to its own experience and that of its neighbors. PSO has been found to be successful in a wide variety of optimization tasks [3.8], but until recently it had not been extended to deal with multiple objectives. PSO seems particularly suitable for multiobjective optimization mainly because of the high speed of convergence that the algorithm presents for single objective optimization [3.8]. Here, we represent, called "multiobjective particle swarm optimization" (MOPSO), which allows the PSO algorithm to be able to deal with multiobjective optimization problems [3.7]. These are precisely the main motivations that led us to apply PSO for multiobjective problems [3.10].

3.3.1 Description of the Algorithm

PSO using a Pareto ranking scheme [3.11] could be the straightforward way to extend the approach to handle multiobjective optimization problems. The historical record of best solutions found by a particle (i.e., an individual) could be used to store nondominated solutions generated in the past (this would be similar to the notion of elitism used in evolutionary multiobjective optimization). The use of global attraction mechanisms combined with a historical archive of previously found nondominated vectors would motivate convergence toward globally nondominated solutions. Flow-chart of the MOPSO is shown in Fig. **??**.

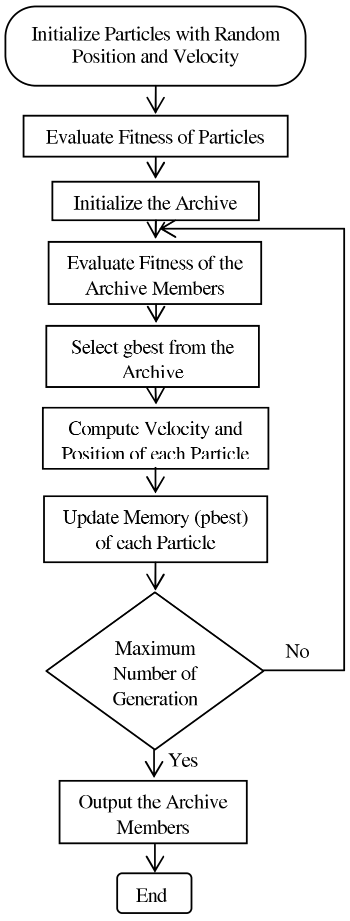

Initialize Particles with Random Position and Velocity
Evaluate Fitness of Particles
Initialize the Archive
Evaluate Fitness of the Archive Members
Select gbest from the Archive
Compute Velocity and Position of each Particle
Update Memory (pbest) of each Particle
Maximum Number of Generation
No
Yes
Output the Archive Members
End

Figure 3.3: MOPSO Flow Chart

3.3.2 Main Algorithm

The algorithm of MOPSO is the following.

1) Initialize the population *pop*.

2) Initialize the speed of each particle *vel*.

3) Evaluate each of the particles in *pop*.

4) Store the positions of the particles that represent nondominated vectors in the repository *rep*.

5) Generate hyper cubes of the search space explored so far, and locate the particles using these hyper cubes as a coordinate system where each particle's coordinates are defined according to the values of its objective functions.

6) Update the memory of each particle (this memory serves as a guide to travel through

the search space. This memory is also stored in the repository).
7) WHILE maximum number of cycles has not been reached.

DO

a) Compute the speed of each particle using the following expression:

$$vel[i] = w * vel[i] + r_1 * (pbest[i] - pop[i]) + r_2 * (rep[h] - pop[i]) \tag{3.2}$$

where w represents inertia weight, r_1 and r_2 are random numbers in the range $[0, 1]$, $pbest[i]$ is the best position that the particle i has had, $rep[h]$ is a global best position that is taken from the repository, the index h is selected in the following way: those hyper cubes containing more than one particle are assigned a fitness equal to the result of dividing any number $x > 1$ by the number of particles that they contain. This aims to decrease the fitness of those hyper cubes that contain more particles and it can be seen as a form of fitness sharing. Then, we apply roulette-wheel selection using these fitness values to select the hypercube from which we will take the corresponding particle. Once the hypercube has been selected, we select randomly a particle within such hypercube. $pop[i]$ is the current value of the particle i.

b) Compute the new positions of the particles adding the speed produced from the previous step

$$pop[i] = pop[i] + vel[i] \tag{3.3}$$

c) Maintain the particles within the search space in case they go beyond their boundaries (avoid generating solutions that do not lie on valid search space). When a decision variable goes beyond its boundaries, then we do two things: 1) the decision variable takes the value of its corresponding boundary (either the lower or the upper boundary) and 2) its velocity is multiplied by (-1) so that it searches in the opposite direction.

d) Evaluate each of the particles in *pop*.

e) Update the contents *rep* of together with the geographical representation of the particles within the hyper cubes. This update consists of inserting all the currently nondominated locations into the repository. Any dominated locations from the repository are eliminated in the process. Since the size of the repository is limited, whenever it gets full, we apply a secondary criterion for retention: those particles located in less populated areas of objective space are given priority over those lying in highly populated regions.

f) When the current position of the particle is better than the position contained in its memory, the particle's position is updated using

$$pbest[i] = pop[i] \tag{3.4}$$

The criterion to decide what position from memory should be retained is simply to apply Pareto dominance (i.e., if the current position is dominated by the position in memory, then the position in memory is kept, otherwise, the current position replaces the one in memory, if neither of them is dominated by the other, then we select one of them randomly).

g) Increment the loop counter

8) END WHILE

3.3.3 External Repository

External repository (or archive) is to keep a historical record of the nondominated vectors found along the search process [3.8].It consists of two main parts: the archive controller and the grid.

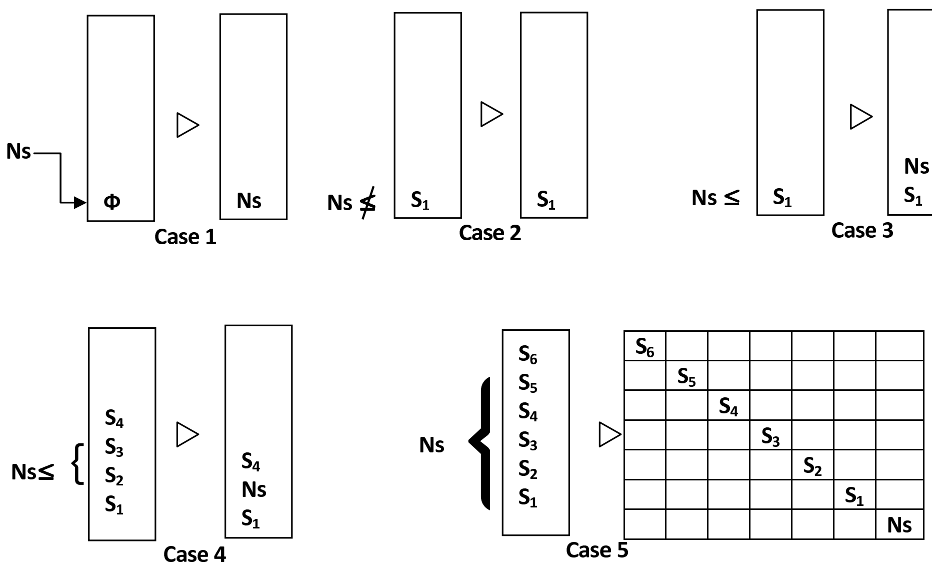


Figure 3.4: Archive Controller

The Archive Controller

In this approach, archive controller is decide whether a certain solution should be added to the archive or not. The decision-making process is the following:

The nondominated vectors found at each iteration in the primary population of our algorithm are compared (on a one-per-one basis) with respect to the contents of the external repository which, at the beginning of the search will be empty. If the external archive is empty, then the current solution is accepted (see case 1, in Fig. ??). If this new solution is dominated by an individual within the external archive, then such a solution is automatically discarded (see case 2, in Fig. ??). Otherwise, if none of the elements contained in the external population dominates the solution wishing to enter, then such a solution is stored in the external archive. If there are solutions in the archive that are dominated by the new element, then such solutions are removed from the archive (see cases 3 and 4, in Fig. ??). Finally, if the external population has reached its maximum allowable capacity, then the adaptive grid procedure is invoked (see case 5, in Fig. ??).

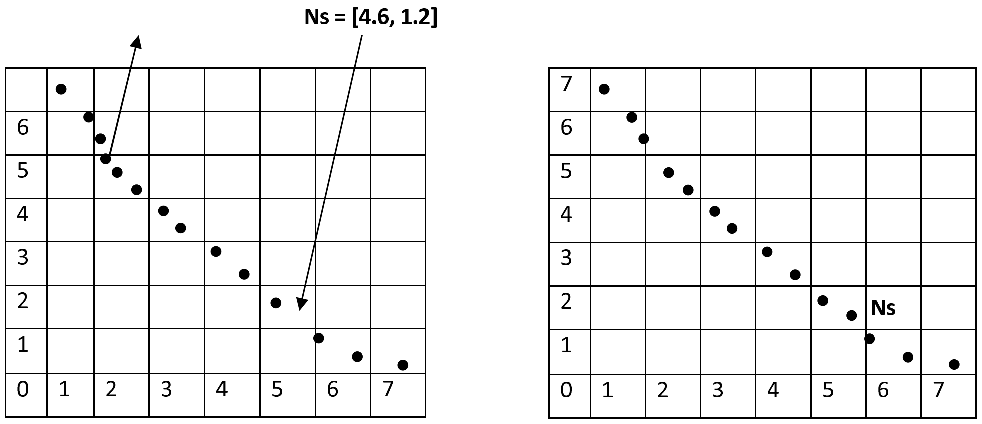


Figure 3.5: Adaptive Grid

The Grid

For the diversity preservation of the solutions, our approach uses a variation of the adaptive grid proposed in [3.8]. All the solutions that are nondominated with respect to the contents of the archive, stored in the external archive. Into the archive, objective function space is divided into regions as shown in Fig. ??. The adaptive grid is a space formed by hyper cubes, have as many components as objective functions. Note that if the individual inserted into the external population lies outside the current bounds of the grid, then the grid has to be recalculated and each individual within it has to be relocated . The main advantage of the adaptive grid is that its computational cost is lower than niching [3.8, 3.11]. In such a case, the computational complexity of the adaptive grid would be the same as niching [i.e., $O(N^2)$].

3.4 Multiobjective Bacteria Foraging Optimization

The social foraging behavior of Escherichia coli bacteria has been used to solve optimization problems [3.12]. In the last decade, approaches based on Bacteria Foraging Optimization (BFO) have received increased attention from the academic and industrial communities for dealing with optimization problems that have been shown to be intractable using conventional problem solving techniques. Natural selection tends to eliminate animals with poor foraging strategies through methods for locating, handling, and ingesting food and favors the propagation of genes of those animals that have successful foraging strategies, since they are more likely to obtain reproductive success [3.13]. After many generations, poor foraging strategies are either eliminated or re-structured into good ones. Since a foraging organism/animal takes actions to maximize the energy utilized per unit time spent foraging, considering all the constraints presented by its own physiology, such as sensing and cognitive capabilities and environmental parameters (e.g., density of prey, risks from predators. physical characteristics of the search area), natural evolution could lead to optimization.

It is essentially this idea that could be applied to Multi-objective optimization problems. The optimization problem search space could be modeled as a social foraging environment where groups of parameters communicate cooperatively for finding solutions to difficult engineering problems.

3.4.1 Bacterial Foraging

Bacteria have the tendency to gather to the nutrient-rich areas by an activity called chemotaxis. It is known that bacteria swim by rotating whip like flagella driven by a reversible motor embedded in the cell wall. E. coli has 8-10 flagella placed randomly on a cell body. When all flagella rotate counterclockwise, they form a compact, helically propelling the cell along a helical trajectory, which is called run. When the flagella rotate clockwise, they pull on the bacterium in different directions, which causes the bacteria to tumble. A brief outline of each of these processes is given in this section.

(1) Chemotaxis: An E. coli bacterium can move in two different ways; it can run (swim for a period of time) or it can tumble, and alternate between these two modes of operation in the entire lifetime. In the BFO, a unit walk with random direction represents a tumble and a unit walk with the same direction in the last step indicates a run. In computational chemotaxis, the movement of the i^{th} bacterium after one step is represented as

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\Delta(j) \tag{3.5}$$

where $\theta^i(j,k,l)$ denotes the i^th bacterium at j^{th} chemotactic, k^{th} reproductive and l^{th} elimination and dispersal. $C(i)$ is the length of unit walk, which is a constant in basic BFO and $\Delta(j)$ is the direction angle of the j^{th} step. When its activity is run, $\Delta(j)$ is same as $\Delta(j-1)$, otherwise, Δ is a random angle directed within a range of $[0, 2\pi]$.

If the cost at $\theta^i(j+1,k,l)$ is better than the cost at $\theta^i(j,k,l)$, then the bacterium takes another step of size $C(i)$ in that direction otherwise it is allowed to tumble. This process is continued until the number of steps taken is greater than the number of chemotactic loop, N_c .

(2) Reproduction: After all N_c chemotactic steps have been covered, a reproduction step takes place. The fitness values of the bacteria are sorted in ascending order. The lower half of the bacteria having higher fitness die and the remaining $S_r = S/2$ bacteria are allowed to split into two identical ones. Thus the population size after reproduction is maintained constant.

(3) Elimination and Dispersal: Since bacteria may stuck around the initial or local optima positions, it is required to diversify the bacteria either gradually or suddenly so that the possibility of being trapped into local minima is eliminated. The dispersion operation takes place after a certain number of reproduction process. A bacterium is chosen, according to a preset probability p_{ed} , to be dispersed and moved to another position within the environment. These events may help to prevent the local minima trapping effectively, but unexpectedly disturb the optimization process. The detailed of this concept is presented in [3.13].

This foraging technique we have applied for the optimization of multiple objectives. Instead of getting single optimal point, in multiobjective optimization large number of nondominated points are the final optimal solutions. The flow chart of the complete algorithm is shown in Fig. **??**.

3.4.2 MOBFO Algorithm

The main goal of the BFO based multi-objective algorithm is to find the non-dominated optimum values of the given functions $J = (f_1, \dots, f_n)$. Here, $\phi^i(j, k, l)$ is the position of a i^{th} bacteria in the population of N at the j^{th} chemotactic step, k^{th} reproduction step, and l^{th} elimination−dispersal event. The main steps of MOBFO given below:

Step 1: Initialize parameters $p, N, N_C, N_s, N_{re}, N_{ed}, P_{ed}, C(i), i = (1, \dots, N), \phi^i$ where,

p: Dimension of the search space,

N: The number of bacteria in the population,

N_C: Chemotactic steps,

N_{re}: The number of reproduction steps,

N_{ed}: The number of elimination−dispersal events,

P_{ed}: Elimination−dispersal with probability,

$C(i)$: The size of the step taken in the random direction specified by the tumble.

Step 2: Elimination−dispersal loop: $l = l + 1$.

Step 3: Reproduction loop: $k = k + 1$.

Step 4: Chemotaxis loop: $j = j + 1$.

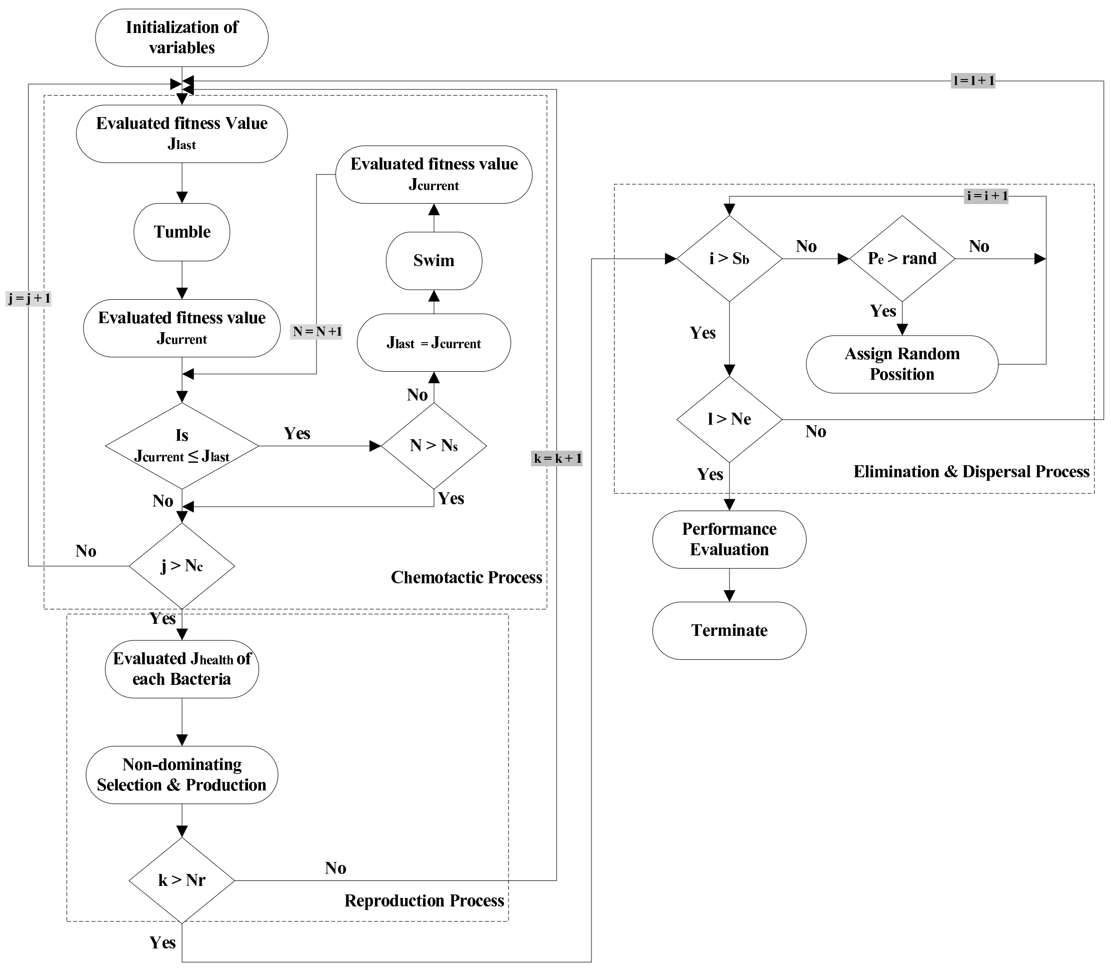


Figure 3.6: MOBFO Flow-chart

[substep a] For $i = 1, 2, ..., N$, take a chemotactic step for bacterium i as follows.

[substep b] Compute fitness function, $J(i, j, k, l)$, where $J = (f_1, f_2, \dots, f_n)$ set of all objective functions.

[substep c] Let $J_{last} = J(i, j, k, l)$ to save this value since we may find a better cost via a run.

[substep d] Tumble: generate a random vector $\Delta(i) \in \Re^p$ with each element $\Delta_m(i), m = 1, 2, \dots, p$, a random number on $[-1, 1]$.

[substep e] Move: Let

$$\phi^i(j+1, k, l) = \phi^i(j, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

This results in a step of size $C(i)$ in the direction of the tumble for bacterium i.

[substep f] Compute $J(i, j+1, k, l)$.

[substep g] Swim.

(i) Let $m = 0$ (counter for swim length).

(ii) While $m < N_s$ (if have not climbed down too long).

Let $m = m + 1$.

If $J(i, j+1, k, l) \preceq J_{last}$ (if dominated), let $J_{last} = J(i, j+1, k, l)$ and let

$$\phi^i(j+1, k, l) = \phi^i(j+1, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

and use this $\phi^i(j+1, k, l)$ to compute the new $J(i, j+1, k, l)$ as we did in [substep f].

Else, let $m = N_s$. This is the end of the while statement.

[**substep h**] Go to next bacterium $(i+1)$ if $i \neq N$ (i.e., go to [substep b] to process the next bacterium).

Step 5: If $j < N_C$, go to step 3. In this case, continue chemotaxis, since the life of the bacteria is not over.

Step 6: Reproduction:

[**substep a**] For the given k and l, and for each $i = 1, 2, ..., N$, let

$$J^i_{health} = J^i_{best}$$

where J^i_{best} represent best fitness value of i^{th} nondominated bacteria according to given objective functions, which is selected randomly from all nondominated Chemotaxis fitness values of i^{th} bacteria.

[**substep b**] Sort bacteria according to nondomination and the crowding distance operators. Less crowded bacteria, in objective space, are selected for the reproduction and chemotactic parameters $C(i)$ is find out with [3.14]

$$C(i) = average\left\{\frac{J^i_{health}}{\left(J^i_{health} + lamda\right)}\right\}$$

where $lamda$ is a constant parameter.

The selected S_r bacteria with the best values split into two bacteria(this process is performed by the copies that are made are placed at the same location as their parent).

Step 7: If $k < N_{re}$, go to [step 3]. In this case, we have not reached the number of specified reproduction steps, so we start the next generation of the chemotactic loop.

Step 8: Elimination−dispersal: For $i = 1, 2, ..., N$, with probability P_{ed}, eliminate and disperse each bacterium, which results in keeping the number of bacteria in the population constant. To do this, if a bacterium is eliminated, simply disperse one to a random location on the optimization domain. If $l < N_{ed}$, then go to [step 2], otherwise end.

3.5 Summary

The attractive feature of multiobjective evolutionary algorithms is their ability to find a wide range of nondominated solutions close to the true Pareto optimal solutions. Because of this advantage, the philosophy of solving multiobjective optimization problems can be revolutionized. Evolutionary algorithms process a population of solutions in each iteration, thereby making them ideal candidates for finding multiple trade-off solutions in one single simulation run. In this chapter, we have presented three multiobjective evolutionary algorithms, which can be used to find multiple nondominated solutions close to the Pareto optimal solutions.

Chapter 4

Performance Evaluation of Multiobjective Evolutionary Algorithms

CHAPTER 4

PERFORMANCE EVALUATION OF MULTIOBJECTIVE EVOLUTIONARY ALGORITHMS

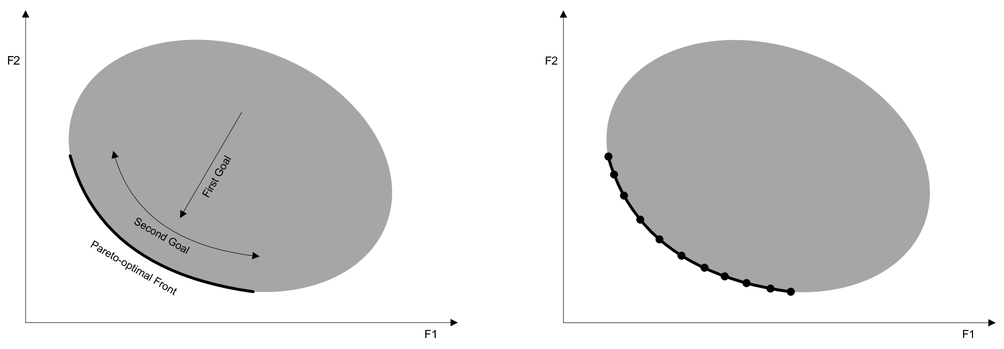
When a new and innovative methodology is initially discovered for solving a search and optimization problem, a visual description is adequate to demonstrate the working of the proposed methodology. MOEA's demonstrated their working by showing the obtained nondominated solutions along with the true Pareto-optimal solutions in the objective space. In these studies, the emphasis has been given to demonstrate how closely the obtained solutions have converged to the true Pareto-optimal front [4.1, 4.2].

4.1 Performance Metrics

With the existence of many different MOEA's , it is necessary that their performance be quantified on a number of test problems. There are two distinct goals in MOO:

- Discover solutions as close to the Pareto-optimal solutions as possible.
- Find solutions as diverse as possible in the obtained nondominated front.

In some sense, these two goals are *orthogonal* to each other. The first goal requires a search *towards* the Pareto-optimal region, while the second goal requires a search *along* the Pareto-optimal front, as depicted in Fig. **??**. The measure of diversity can also be separated in two different measures of *extent* (mean along the spread of extreme solutions) and distribution (meaning the relative distance among solutions).



(a) Two goals for multiobjective optimization (b) Ideal non-dominated solutions

Figure 4.1: Nondominated Pareto Front

Many performance metrics have been suggested [4.1-4.4]. Here, we define two performance metrics that are more direct in evaluating each of the above two goals in a solution set obtained by a multiobjective optimization algorithm.

4.1.1 Convergence Metric

The first metric Convergence Metric 'γ' measures the extent of convergence to a known set of Pareto-optimal solutions [4.3]. Since multiobjective algorithms would be tested on problems having a known set of Pareto-optimal solutions, the calculation of this metric is possible. This metric explicitly computes a measure of the closeness of a set Q of N solutions from a known set of the True Pareto-optimal set P^*. Convergence Metric finds an average distance of Q from P^*, as follows

$$\gamma = \frac{\sum_{i=1}^{N} d_i}{N} \tag{4.1}$$

where the parameter d_i is the Euclidean distance (in the objective space) between the solution $i \in Q$ and the nearest member of P^*:

$$d_i = \overbrace{\underbrace{min}_{k=1}}^{|P^*|} \sqrt{\sum_{m=1}^{M} (f_m^{(i)} - f_m^{*(k)})^2} \tag{4.2}$$

where $f_m^{*(k)}$ is the m^{th} objective function value of the k^{th} member of P^*.

When all obtained solutions lie exactly on P^* chosen solutions, this metric takes a value of zero. In all simulations performed here, we present the average $\overline{\gamma}$ and σ_γ variance of this metric calculated for solution sets obtained in multiple runs.

4.1.2 Diversity Metric

Although above metric alone can provide some information about the spread in obtained solutions, Deb [4.3] suggested different metric, called Diversity Metric, to measure the spread in solutions obtained by an algorithm directly. The Diversity Metric 'Δ' measures the extent of spread achieved among the obtained solutions given by:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{N} |d_i - \overline{d}|}{\sum_{m=1}^{M} d_m^e + N\overline{d}} \tag{4.3}$$

where d_i an be any distance measure between neighboring solutions and $\overline{d}$ is the mean value of these distance measure. The parameter d_m^e is distance between the extreme solutions of P^* and Q corresponding to m^{th} objective function. However, a good distribution would make all distances d_i equal to $\overline{d}$ and would make $d_f = d_l = 0$ (with existence of extreme solutions in the nondominated set). Thus, for the most widely and uniformly spread out set of nondominated solutions, the numerator Δ of would be zero, making the metric to take a value zero. For any other distribution, the value of the metric would be greater than zero.

4.2 Multiobjective Test Problems

In multi-objective evolutionary computation, researchers have used many different test problems with known sets of Pareto-optimal solutions. For the comparison of the multiobjective algorithms, we choose popular test functions. These test problems are given below.

4.2.1 Schaffer's (SCH) problem

It is proposed by J. D. Schaffer [4.5].

$$SCH_{Minimize} : \begin{cases} f_1(\mathbf{x}) = x^2, \\ \\ f_2(\mathbf{x}) = (x-2)^2, \\ \\ -1000 \leq x \leq 1000. \end{cases}$$

4.2.2 Kursave(KUR) problem

Our second test function was proposed by Kursawe [4.6].

$$KUR_{Minimize}: \begin{cases} f_1(\mathbf{x}) = \sum_{i=1}^{2}[-10\, exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})], \\ \\ f_2(\mathbf{x}) = \sum_{i=1}^{3}[|x_i|^{0.8} + 5sin(x_i^3)], \\ \\ -5 \leq x_i \leq 5,\ i = 1,2,3. \end{cases}$$

4.2.3 DEB-1 problem

This test problem proposed by the K. Deb. [4.7].

$$Deb-1_{Minimize}: \begin{cases} f_1(\mathbf{x}) = x_1, \\ \\ f_2(\mathbf{x}) = g(x_2)/x_1, \\ where\ g(x_2) = 2.0 - exp\left\{-(\frac{x_2-0.2}{0.004})^2\right\} - 0.8\, exp\left\{-(\frac{x_2-0.6}{0.4})^2\right\} \\ \\ 0.1 \leq x_1, x_2 \leq 1.0. \end{cases}$$

4.2.4 DEB-2 problem

It is proposed by K.Deb [4.7].

$$Deb-2_{Minimize}: \begin{cases} f_1(\mathbf{x}) = x_1, \\ \\ f_2(\mathbf{x}) = g(x_1,x_2).h(x_1,x_2), \\ \\ where \\ g(x_1,x_2) = 11 + x_2^2 - 10.cos(2\pi x_2), \\ h(x_1,x_2) = 1 - \sqrt{\frac{f_1(x_1,x_2)}{g(x_1,x_2)}} \quad if\ f_1(x_1,x_2) \leq g(x_1,x_2), \quad otherwise\ \ 0 \\ \\ 0 \leq x_1 \leq 1, -30 \leq x_2 \leq 30. \end{cases}$$

4.3 Comparison of Multiobjective Evolutionary Algorithms

With the availability of many multiobjective evolutionary algorithms, it is natural to ask which of them (if any) performs better when compared to other algorithms on various test problems. Here, we compare mainly three algorithms, NSGA-II, MOPSO and proposed

MOBFO, for given test problems which are commonly used. For the following test problems the NSGA-II was run using a population size of 100, a crossover rate of 0.8 (uniform crossover was adopted), tournament selection, and a mutation rate of 0.1. MOPSO used a population of 100 particles, a repository size of 100 particles, a mutation rate of 0.5 and 30 divisions for the adaptive grid. Various parameters used in the simulation study for MOBFO are $S_b = 100$, $N_s = 2$, $N_c = 4$, $N_{re} = 10$, $N_{ed} = 2 - 5$, $P_{ed} = 0.1$, $C(i) = 0.05$ and $lamda = 400$. These parameters were kept for all the test problems and we only changed the total number of fitness function evaluations but the same value was adopted for all the algorithms in each of the test problem presented next. In all the following test problems, we report the results obtained from performing 100 independent runs of each algorithm compared.

4.3.1 SCH Problem

In this example, the total number of fitness function evaluations was set to 10,000.

Fig. **??** show the graphical results produced by our MOBFO,the NSGA-II, and MOPSO in the SCH test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables **??**, **??** and **??** shows the comparison of results among the three algorithms considering the metrics previously described. It can be seen that the average performance of MOBFO is slightly below NSGA-II and better than MOPSO with respect to the diversity metric. With respect to convergence metric it places slightly below the NSGA-II and the MOPSO, but with best value than the MOPSO. Also, it is important to notice the very high speed of MOBFO, which requires almost one third of the time than the MOPSO and one tenth of the time than the NSGA-II in this test problem.

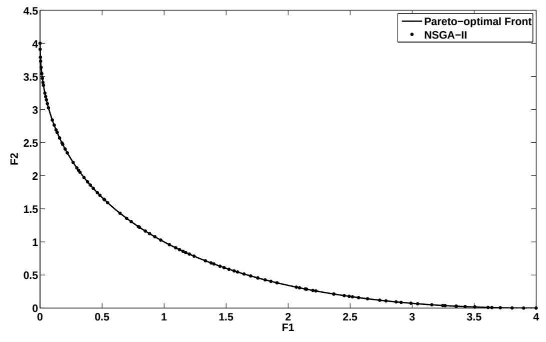
Computational Time	NSGA-II	MOPSO	MOBFO
Best	15.40	4.98	1.51
Worst	26.64	6.26	1.96
Mean	17.91	5.57	1.78
Variance	5.6329	0.0820	0.0091

Table 4.1: Computational Time (in sec) required by each Algorithm for the SCH Test Function

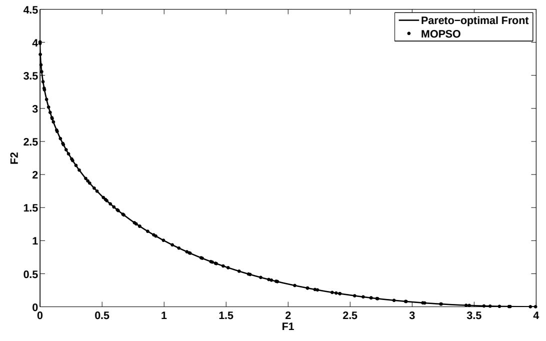
4.3.2 KUR Problem

In this example, the total number of fitness function evaluations was set to 20,000.

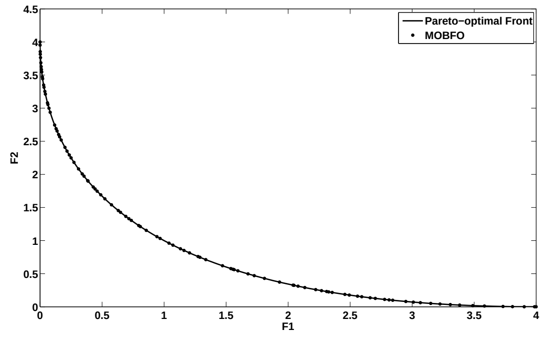
Fig. **??** show the graphical results produced by NSGA-II, the MOPSO, and our MOBFO in the KUR test function chosen. The true Pareto front of the problem is shown



(a) NSGA-II



(b) MOPSO



(c) MOBFO

Figure 4.2: Nondominated Pareto Front for SCH problem

Convergence Metric	NSGA-II	MOPSO	MOBFO
Best	0.0066	0.0070	0.0068
Worst	0.0089	0.0091	0.0095
Mean	0.0079	0.0079	0.0081
Variance	1.79E-07	1.96e-07	2.31E-07

Table 4.2: Results of the Convergence Metric for the SCH Test Function

as a discontinuous line. Tables **??**, **??** and **??** show the comparison of results among the

Diversity Metric	NSGA-II	MOPSO	MOBFO
Best	0.3467	0.5112	0.4379
Worst	0.5140	0.7168	0.6425
Mean	0.4243	0.5938	0.5477
Variance	0.0012	0.0015	0.0019

Table 4.3: Results of the Diversity Metric for the SCH Test Function

three algorithms considering the metrics previously described. It can be seen that the average performance of MOBFO is considerably worse with respect to the convergence and diversity metric, but it have best value of these metric in comparison to others. Also, note that MOBFO is 8 times faster than the NSGA-II in this test function.

Computational Time	NSGA-II	MOPSO	MOBFO
Best	28.46	2.32	3.26
Worst	40.15	3.21	4.09
Average	29.63	2.43	3.73
Variance	5.1621	0.0082	0.0345

Table 4.4: Computational Time (in sec) required by each Algorithm for the KUR Test Function

Convergence Metric	NSGA-II	MOPSO	MOBFO
Best	0.0449	0.0515	0.0349
Worst	0.0559	0.0725	0.2968
Average	0.0516	0.0608	0.0634
Variance	5.72E-06	1.51E-05	9.10E-04

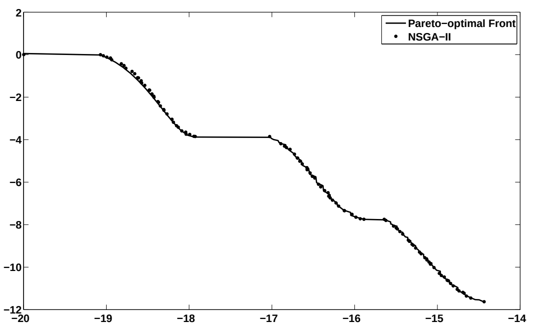
Table 4.5: Results of the Convergence Metric for the KUR Test Function

Diversity Metric	NSGA-II	MOPSO	MOBFO
Best	0.7248	0.6800	0.7203
Worst	0.7939	0.7582	0.8648
Average	0.7597	0.7193	0.7709
Variance	0.0002	0.0003	0.0004

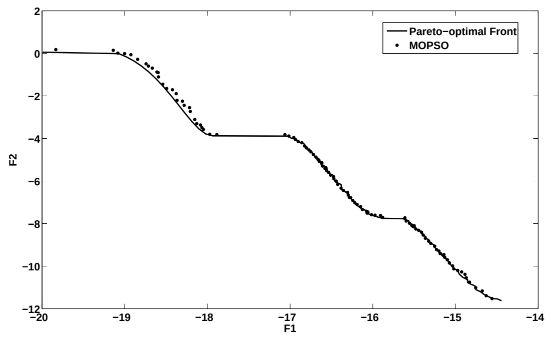
Table 4.6: Results of the Diversity Metric for the KUR Test Function

4.3.3 DEB-1 Problem

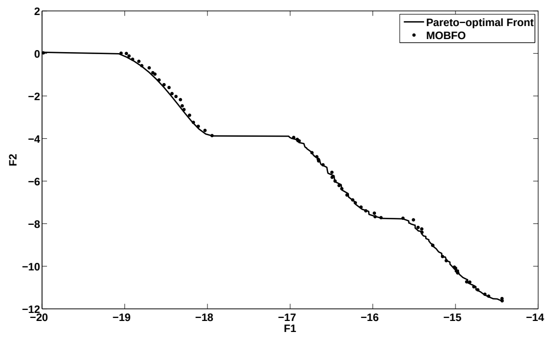
In this example, the total number of fitness function evaluations was set to 15,000.



(a) NSGA-II



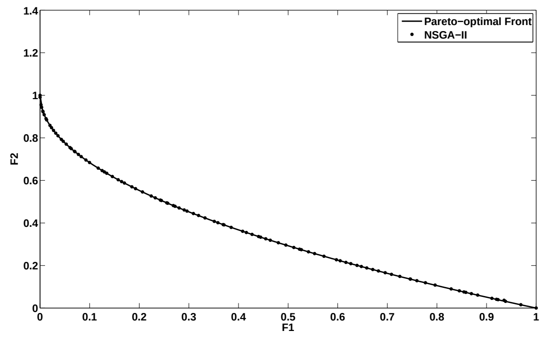
(b) MOPSO



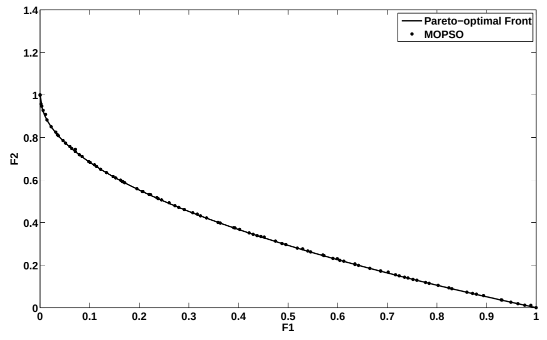
(c) MOBFO

Figure 4.3: Nondominated Pareto Front for KUR problem

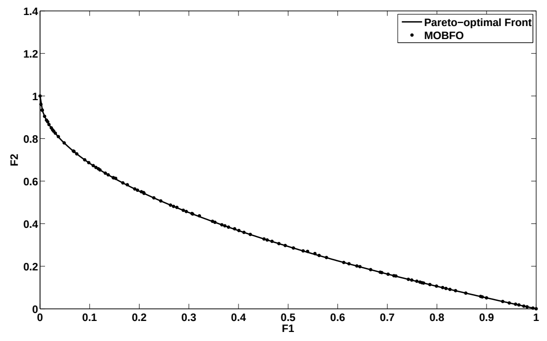
Fig. **??** show the graphical results produced by our MOBFO, the NSGA-II, and MOPSO in the DEB-1 test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables **??**, **??** and **??** show the comparison of results among the three algorithms considering the metrics previously described. It can be seen that the average performance of MOBFO does considerably worse than the NSGA-II and MOPSO with respect to convergence, but plays second with respect to the diversity metric. With respect to computational time, MOBFO is 30 times faster than the NSGA-II and 1.5 times faster than MOPSO in this test function.



(a) NSGA-II



(b) MOPSO



(c) MOBFO

Figure 4.4: Nondominated Pareto Front for DEB1 problem

Computational Time	NSGA-II	MOPSO	MOBFO
Best	20.56	0.95	0.64
Worst	29.26	2.09	1.85
Average	21.47	1.37	1.55
Variance	2.0839	0.0429	0.070

Table 4.7: Computational Time (in sec) required by each Algorithm for the DEB-1 Test Function

Convergence Metric	NSGA-II	MOPSO	MOBFO
Best	0.0021	0.0021	0.0022
Worst	0.0041	0.0034	0.5301
Average	0.0028	0.00269	0.0083
Variance	1.57E-07	6.54E-08	0.0028

Table 4.8: Results of the Convergence Metric for the DEB-1 Test Function

Diversity Metric	NSGA-II	MOPSO	MOBFO
Best	0.3344	0.4803	0.4242
Worst	0.7825	0.6413	0.8911
Average	0.4399	0.5602	0.5558
Variance	0.0059	0.0013	0.0060

Table 4.9: Results of the Diversity Metric for the DEB-1 Test Function

4.3.4 DEB-2 Problem

In this example, the total number of fitness function evaluations was set to 25,000.

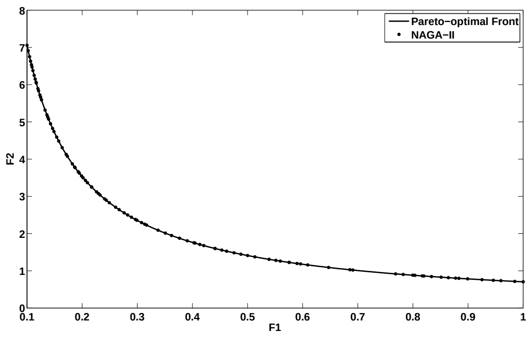
Fig. 4.5 show the graphical results produced by the NSGA-II, the MOPSO and our MOBFO in the DEB-2 test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables **??**, **??** and **??** show the comparison of results among the three algorithms considering the metrics previously described. It can be seen that the average performance of MOBFO is the better than the MOPSO but slightly below NSGA-II with respect to the diversity metric, and it is better than the NSGA-II but slightly below MOPSO and have best value with respect to the convergence metric. Again, it is 10 times faster than the NSGA-II for this test problem.

Computational Time	NSGA-II	MOPSO	MOBFO
Best	30.64	0.75	3.31
Worst	46.48	4.62	4.34
Average	33.82	2.09	3.98
Variance	16.3501	0.4590	0.0366

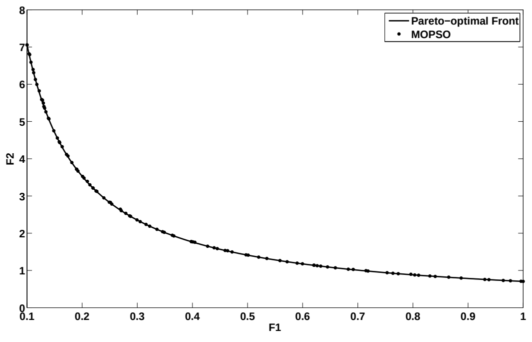
Table 4.10: Computational Time (in sec) required by each Algorithm for the DEB-2 Test Function

4.4 Conclusion

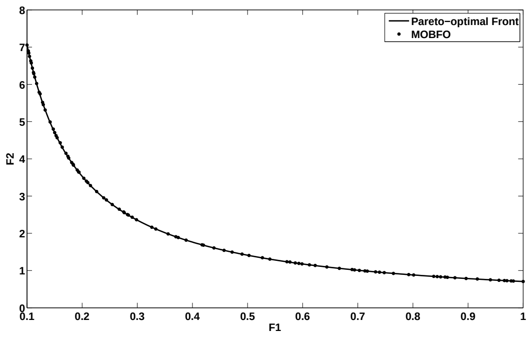
We have presented a proposal to extend BFO to handle multiobjective problems. The proposed algorithm is relatively easy to implement and it improves the exploratory capabilities of BFO by introducing adaptive step size whose range of action varies over



(a) NSGA-II



(b) MOPSO



(c) MOBFO

Figure 4.5: Nondominated Pareto Front for DEB2 problem

Convergence Metric	NSGA-II	MOPSO	MOBFO
Best	0.0148	0.0093	0.0093
Worst	0.9578	0.1569	1.3668
Average	0.2096	0.0259	0.0607
Variance	0.1236	0.0008	0.0378

Table 4.11: Results of the Convergence Metric for the DEB-2 Test Function

time. The results indicate that our approach is a viable alternative since it has an av-

Diversity Metric	NSGA-II	MOPSO	MOBFO
Best	0.5104	0.6947	0.5136
Worst	0.7904	1.3575	1.0059
Average	0.6425	0.8582	0.6983
Variance	0.0053	0.0282	0.0102

Table 4.12: Results of the Diversity Metric for the DEB-2 Test Function

erage performance highly competitive with respect to other multiobjective evolutionary algorithms.

The exceptionally low computational times required by our approach make it a very promising approach to problems in which the computational cost is a vital issue (e.g., engineering optimization). One aspect that we would like to explore in the future is the use of a some modified crowding operator to improve the distribution of nondominated solutions along the Pareto front. This would improve the capabilities of the algorithm to distribute uniformly the nondominated vectors found. We are also considering the possibility of extending this algorithm so that it can deal with dynamic functions. Finally, it is desirable to study in more detail the parameters fine tuning required by the algorithm, as to provide a more solid basis to define them.

Chapter 5

Application : Radar Pulse Compression

CHAPTER 5

APPLICATION : RADAR PULSE COMPRESSION

5.1 Introduction

Radar is a contraction of the words *radio detection and ranging.* Radar is an electromagnetic system for the detection and location of objects such as aircraft, ships, spacecraft, vehicles, people, and natural environment. It operates by transmitting a particular type of waveform, a pulse modulated sine wave example, in the space and detecting the the echo signal reflected from the target.

The basic principle of radar is simple [5.1 - 5.2]. A transmitter generates an electromagnetic signal (such as a short pulse of sine wave) that is radiated into space by an antenna. A portion of the transmitted signal is intercepted by a reflecting object (target) and is re-radiated in all directions. It is the energy re-radiated in back direction that is of prime interest to the radar. The receiving antenna collects the returned energy and delivers it to a receiver, where it is processed to detect the presence of the target and to extract its location and relative velocity. The distance to the target is determined by measuring the time taken for the radar signal to travel to the target and back. The range is

$$R = \frac{cT_R}{2} \tag{5.1}$$

where T_R is the time taken by the pulse to travel to target and return, c is the speed of propagation of electromagnetic energy (speed of light). A radar provides the good range resolution as well as long detection of the target.

High range resolution, as might be obtained with a short pulse, is important for many radar applications. There can be limitations, however, to the use of a short pulse. Since the spectral bandwidth of a pulse is inversely proportional to its width, the bandwidth of a sort pulse is large. Large bandwidth increase system complexity, make greater demands on the signal processing and increase the likelihood of interference to and from other users of the electromagnetic spectrum. Wide bandwidth can also mean less dynamic range in the receiver because receiver noise power is proportional to bandwidth. Also, a short pulse waveform provides less accurate radial velocity measurement than if obtained from the Doppler-frequency shift.

A serious limitation to achieving long range with short-duration pulses is that a high peak power is required for a large pulse energy. The transmission line of a high peak power radar can be subject to voltage breakdown, especially at the higher frequencies where waveguide dimensions are small. If the peak power is limited by breakdown, the pulse might not have sufficient energy for long detection.

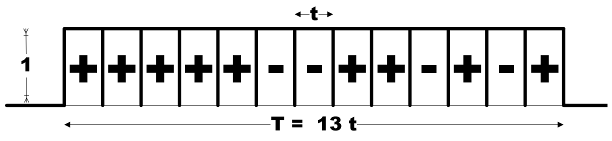
So that there are trade-off between the range resolution and long range detection for a given length of pulse. To solve the dilemma between these two, Pulse Compression technique is used in the pulse radar detection [5.1-5.2]. The technique of using a long, modulated pulse to obtain the range resolution of a sort pulse, but with the energy of a long pulse, is known as *pulse compression.*

5.2 Pulse Compression

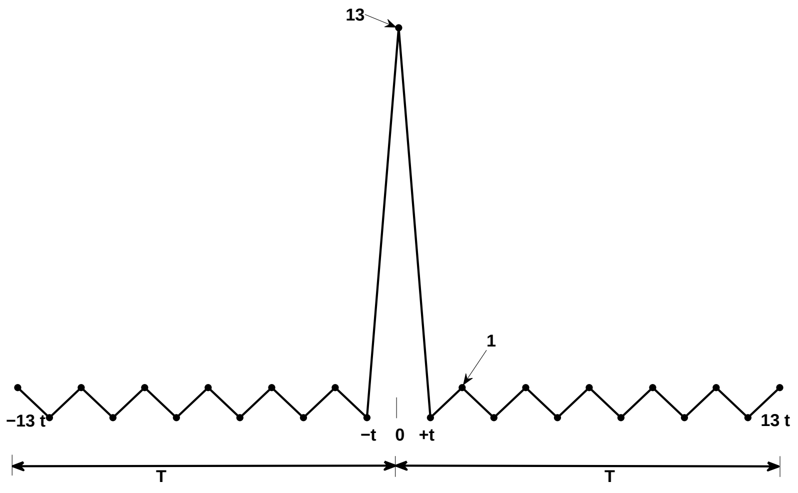
Pulse compression is a signal processing technique mainly used in radar and sonar to augment the range resolution as well as the signal to noise ratio. Pulse compression allow a radar to utilize a long pulse to achieve large radiated energy, but simultaneously to obtain the range resolution of a short pulse. It accomplishes this by employing frequency or phase modulation to widen the signal bandwidth. A short pulse has a wide spectral bandwidth. A long pulse of width T can be have the same spectral bandwidth as a short pulse if the long pulse is modulated in frequency or phase. The modulated long pulse with its increased bandwidth $B(>> 1/T)$ is compressed by the matched filter of the receiver to a width equal to $\tau \approx 1/B$. This achieves the benefit of high range resolution with out the need to resort to a short pulse, where B is the modulated-phase spectral bandwidth. Pulse compression is attractive when the peak power required of a short-pulse radar cannot be achieved with practical transmitters.

Pulse compression is a method for achieving most of the benefits of a short pulse while keeping within the practical constraints of the peak-power limitation. There are many types of modulations used for pulse compression, but one that seen wide application is the phase-coded pulse.

Phase-coded pulse compression: In this form of pulse compression, a long pulse of duration T is divided into N sub-pulses each of width τ as shown in Fig. **??**. An increase in bandwidth is achieved by changing the phase of each sub-pulses. The phase of each sub-pulse is chosen to be either 0 or π radians. The output of the matched filter will be a spike of width τ with an amplitude N times greater than that of long pulse [5.2]. The pulse-compression ratio is $N = T/\tau \approx BT$, where $B \approx 1/\tau = bandwidth$. The output waveform extends a distance T to either side of the peak response, or central spike. The portions of the output waveform other than the spike are called *time side-lobes.*



(a) 13 element Barker Code



(b) Auto-correlation Output

Figure 5.1: Pulse Compressed Signal

Barker codes: The binary choice of 0 or π phase for each sub-pulse may be made at random. However, some random selections may be better suited than others for radar application. One criterion for the selection of a good "random" phase-coded waveform is that its autocorrelation function should have equal time side-lobes. The binary phase-coded sequence of 0, π values that result in equal side-lobes after passes through the matched filter is called a *Barker code.* An example shown in Fig. **??**. This is a Barker code of length 13. The (+) indicates 0 phase and (−) indicates π radians phase. The auto-correlation function, or output of the matched filter, is shown in Fig **??**. There are six equal time side-lobes to either side of the peak, each of label −22.3 dB below the peak. The longest Barker code length is 13. When a larger pulse-compression ratio is desired, some form of pseudo random code is usually used. To achieve high range resolution without an incredibly high peak power, one needs pulse compression.

Matched filter: Under certain conditions maximizing the output peak-signal to noise (power) ratio of a radar receiver maximizes the detectability of a target. A linear network that does this is called a *matched filter*. It has a frequency response function which is proportional to the complex conjugate of the signal spectrum.

$$H(f) = G_a S^*(f)\ exp(-j2\pi f t_m) \tag{5.2}$$

where G_a is a constant, t_m is the time at which the output of the matched filter is a maximum (generally equal to the duration of the signal), and $S^*(f)$ is the complex conjugate of the spectrum of the (received) input signal $s(t)$, found from the Fourier transform of the received signal $s(t)$ such that

$$S(f) = \int_{-\infty}^{\infty} s(t)\ exp(-j2\pi f t)\ dt \tag{5.3}$$

A matched filter for a transmitting a rectangular shaped pulse is usually characterized by a bandwidth B approximately the reciprocal of the pulse with τ or $B\tau \approx 1$. The output of a matched filter receiver is the cross-correlation between the received waveform and a replica of the transmitted waveform.

5.3 Pulse Compression Using Particle Swarm Optimization

5.3.1 Particle swarm optimization (PSO)

The PSO is motivated from the simulation of social behavior [5.5]. In evolutionary computational algorithms, evolutionary operators are used to manipulate the individuals. But in PSO these individuals are evolved through generations by cooperation and competition among the individuals. Each individual in PSO flies in the search space with a velocity which is dynamically adjusted according to its own as well as its companions' flying experiences. Each individual is treated as a volume-less particle in a D-dimensional space.

The i^{th} particle is represented as $X_i = [x_{i1}, x_{i2}, \dots, x_{iD}]^T$. The position giving the best previous value is called the best previous position of any particle and is represented as $P_i = [p_{i1}, p_{i2}, \dots, p_{iD}]^T$. The rate of change of position for i^{th} particle is represented as $V_i = [v_{i1}, v_{i2}, \dots, v_{iD}]^T$. The overall best location (global) obtained so far by the particles is also tracked by the optimizer and is represented by $P_g = [p_{g1}, p_{g2}, \dots, p_{gD}]^T$. The velocity and position of the d^{th} element of i^{th} particle at $(k+1)^{th}$ search from the knowledge of previous search are changed according to (5.4) and (5.5) respectively.

$$v_{id}(k+1) = H(k) * v_{id}(k) + c_1 * r_1 * (p_{id}(k) - x_{id}(k)) + c_2 * r_2 * (p_{gd}(k) - x_{id}(k)) \quad (5.4)$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (5.5)$$

The acceleration constants c_1 and c_2 in (5.4) represent the weighting of the stochastic acceleration terms that pull each particle towards their best and global best positions. Low values allow particles to roam far from target regions while high values results in abrupt movement towards or past target regions. The acceleration constants are usually taken to be 2.0 for almost all applications. The inertia weight, H has characteristics that are reminiscent of temperature parameter in the simulated annealing. A large inertia weight facilitates a global search while a small inertia weight facilitates a local search. By linearly decreasing the inertia weight from a large value (close to unity) to a small value through the course of PSO run, the PSO tends to have more global search ability at the beginning of the run while possessing more local search ability near the end of the run. The inertia weight is chosen [6.22] according to the relation

$$H(k) = H_0 * \frac{(H_o - H_1) * k}{I} \quad (5.6)$$

where k = search number,

I = maximum number of iterations,

The PSO is similar to a genetic algorithm (GA) in that the system is initialized with a population of random solution. But unlike in GA, each potential solution in PSO is assigned a randomized velocity and the potential solutions, called particles are then flown through the problem space. Each particle keeps track of its coordinates in the problem space which is associated with the best solution (fitness) it has achieved so far and is denoted by P_i.

The steps in implementing the PSO algorithms are:

1. Initialize a population of particles with random positions and velocities on D-dimensions in the problem space (the problem consists of D variables).

2. For each particle, evaluate the fitness function in D variables.

3. Compare the fitness value of each particle with that obtained from the personal best value. If the current position, X_i provides better fitness value than provided by P_i, then it is made equal to the current value and the previous location is updated to the current location in D-dimensional space.

4. Compare each fitness value with that given by overall best value,P_g of the population. If the current value is better than that offered by P_g then reset P_g to the current best position vector.

5. Update the velocity and position of each particle according to (5.4) and (5.5) respectively.

6. Loop to step (2) until a pre-specified criterion is met for example the minimum mean square error (MMSE) is achieved or all the particles attain a common best position.

5.3.2 Problem Formulation

In this section, the optimal weights of the mismatch filter is find out with the help of evolutionary technique, called Particle Swarm Optimization (PSO). A constraint optimization problem is formed for reduction of the side-lobe levels. The constrained optimization problem is usually written as [5.6]

$$\begin{aligned} & Maximize\ f(\vec{x}) \\ Subject\ to\ \ & g_j(\vec{x}) \leq 0, \quad j = 1, 2, ..J \\ & h(\vec{x}) = 0, \quad k = 1, 2, ..K \end{aligned} \tag{5.7}$$

In the above problem, $f(\vec{x})$ is the objective function. There are n variables $\vec{x} = [x_1, x_2, x_3, \ldots, x_n]$, J less than equal to type inequality constraints $g(\vec{x})$ and K equality constraints $h(\vec{x})$.

A single pulse whose carrier is phase coded, is often used in pulse radar system to increase range resolution and accuracy when energy requirements dictate a pulse length substantially in excess of the desired resolution. For a given binary sequence

$$S = \{s_0 = \pm 1, s_1, \ldots, s_{N-1}\}$$

where N is the number samples of binary sequence. The output of the matched filter is the autocorrelation, whose values are given by

$$C_k(S) = \sum_{i=0}^{N-|K|-1} s_i \ s_{i+|k|} \tag{5.8}$$

for $-(N-1) \leq k \leq (N-1)$

For the mismatch filter formulation, the filter elements are taken as

$$H = \{h_0, h_1, \ldots, h_{M-1}\}$$

where the weights are real and $M(\geq N)$ shows the filter length. For both symmetry and mathematical convenience M will be assumed to be odd if N is odd and even if N is even. The output of the mismatch filter are given by

$$B_k(S,H) = \sum_{i=0}^{M-1} h_i \ s_{i-k} \tag{5.9}$$

for $-(N-1) \leq k \leq (M-1)$

Here we assume $s_i = 0$ if $i < 0$ or $i > N-1$. The constraint function is defined as

$$\begin{aligned} Maximize_H \ f &= B_k|_{k=(\frac{M-N}{2})} \\ &= \sum_{i=0}^{M-1} h_i s_{i-(\frac{M-N}{2})} \\ Subject\ to \left| \sum_{i=0}^{M-1} h_i \ \ s_{i-k} \leq 1 \right| & \end{aligned} \tag{5.10}$$

for $-(N-1) \leq k \leq (M-1)$, $k \neq \frac{M-N}{2}$

where f is the objective function, which shows main lobe level, which to be maximized. The inequality condition restricts the all side-lobes to be less than and equal to one in absolute value. After optimization process, with the help of PSO, the solutions of the swarm are the optimized weights of the mismatched filter.

5.3.3 Performance Evaluation

This section illustrates the performance of the constraint based mismatch filter with the ACF and LS method. In all case we consider the 13-element Barker code and 35-element combined Barker code. For the least square method, filter length is taken same as length of input barker sequence. Parameters of the PSO for the optimization process is taken as

Population = 200

Maximum iteration = 1000
$C_1 = C_2 = 0.7$
$H_0 = 0.4$, $H_1 = 0.9$

Signal to Side-lobe ratio (SSR)

The SSR is defined as the ratio of the peak signal amplitude to maximum side-lobe amplitude. The results of the investigation are depicted in Table **??**. It shows that the proposed constraint based mismatch filter approach achieved higher output SSR compared to other approaches.

Algorithms	SSR (dB)	
	13-element Barker code	35-element Barker code
ACF	22.27	13.97
LS	24.00	16.61
PSO	25.46	18.05

Table 5.1: PSO : Comparison of SSR (in dB)

Noise Performance

Noise is an unwanted random signal which interferences with target echo. If the noise is large enough, it can mask altogether the true target echo so that performance examining the noise rejection ability of Pulse Compression can not be neglected. The input signals used to evaluate the noise robustness are by 13-elements barker code and 35-elements combined barker code and both of them are perturbed by the additive white Gaussian noise (AWGN) with different SNR $(5, 20, 40, 60)$, respectively. The noise performance comparison results of the ACF, LS and constraint based mismatch filter on SSR with different noise environments are shown in the Table **??** for 13-element Barker code and in Table **??** for 35-element combined Barker code.

Algorithms	SSR (dB)			
	SNR = 5 dB	20 dB	40 dB	60 dB
ACF	12.25	19.39	22.03	22.24
LS	14.18	20.61	23.94	23.99
PSO	14.87	20.89	25.28	25.44

Table 5.2: PSO : Comparison of SSR (in dB) for 13 element Barker code for different noisy condition

Algorithms	SSR (dB)			
	SNR = 5 dB	20 dB	40 dB	60 dB
ACF	10.22	13.36	14.00	13.97
LS	11.67	16.34	16.68	16.60
PSO	12.76	16.91	17.79	18.03

Table 5.3: PSO : Comparison of SSR (in dB) for 35 element Combined Barker code for different noisy condition

5.3.4 Conclusion

This section has proposed a novel PSO algorithm for the Pulse Compression in radar system. Peak side-lobe level, SSR with the help of PSO is given better result than the LS and ACF. In this section, it is shown that PSO performs well in minimizing the side-lobes in different noise conditions. The success is due to the formulation of the constraint problem in comparison to the LS, where least square method is used. Simulations have demonstrated that the side-lobe level at the output of mismatch filter can be significantly decreased.

5.4 Development of Functional Link Artificial Neural Network (FLANN) Model for Radar Pulse Compression

5.4.1 FLANN Structure

Pao originally proposed FLANN and it is a novel single layer ANN structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries [5.7,5.8]. Here, the initial representation of a pattern is enhanced by using nonlinear function and thus the pattern dimension space is increased. The functional link acts on an element of a pattern or entire pattern itself by generating a set of linearly independent function and then evaluates these functions with the pattern as the argument. Hence separation of the patterns becomes possible in the enhanced space. The use of FLANN not only increases the learning rate but also has less computational complexity [5.9]. However, it is structurally simple and involves less computations compared to those of MLANN [5.10]. The nonlinear input is generated by functionally expanding the input vector in a nonlinear manner. Different nonlinear expansions may be employed. These are trigonometric (sine and cosine), Chebyshev and power series. In this chapter the trigonometric expansion based FLANN model is developed for pulse compression as it offers better performance compared to when other expansions are used. The proposed model consists of three basic processes.

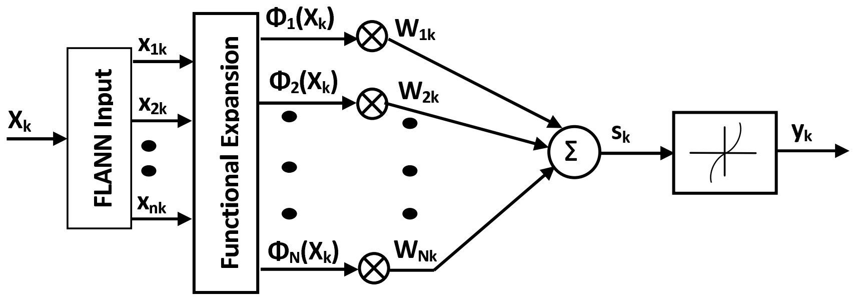


Figure 5.2: FLANN Structure

The proposed FLANN model for Pulse Compression is shown in Fig. **??**, where the block labeled ‘Functional Expansion’ denotes a functional expansion. Here we use a set of N basis functions $\psi(X_k) = [\phi_1(X_k), \phi_2(X_k), \ldots, \phi_N(X_k)]^T$ to functionally expand the input signals $X_k = [x_{1k}, x_{2k}, \ldots, x_{nk}]^T$. These N linearly independent functions map the n-dimensional signal space into an N-dimensional space, that is $\Re^n \rightarrow \Re^N$ for $n < N$. This mapping transforms the linearly non separable problems in the original low-dimensional signal space into separable one in a high-dimensional space.

Several mapping functions have been used to achieve this purpose, such as Legendre, Chebyshev, and trigonometric polynomials. Among these choices the Fourier series being composed of the linear combination of trigonometric harmonics is generally recognized as one of the best approximating functions. A set of basis functions for the trigonometric polynomial Fourier series can be written as

$\{1, cos(\pi u), sin(\pi u), cos(2\pi u), sin(2\pi u), \ldots, cos(N\pi u), sin(N\pi u)\}$. The linear combination of these function values can be presented in its matrix form,

$$S_k = \psi(X_k)\mathbf{W}_k \tag{5.11}$$

where $\mathbf{W}_k = [w_{1k}, w_{2k}, \ldots, w_{Nk}]^T$ is the $1 \times N$ dimensional weighting matrix. The matrix S_k is fed into a bank of identical nonlinear functions to generate the equalized output y_k

$$y_k = \rho(S_k) \tag{5.12}$$

Here the nonlinear function is defined as $\rho(.) = logh\,(.)$.

5.4.2 Learning Algorithm for FLANN Structure

During training process, each expanded input pattern $X_k = [x_{1k}, x_{2k}, \ldots, x_{nk}]^T$ is applied to the model sequentially and the desired compressed value is supplied at the output. Given the input, the model produces an output d_k which acts as an estimate of the desired value,which is 1 when all element of code is present at input [5.11]. The output of the linear part of the model is computed as

$$\begin{aligned} S_k &= \sum_{i=1}^{N} w_{ik}\ \phi_i(X_k) \\ &= \mathbf{W}_k\ \psi(X_k) \end{aligned} \tag{5.13}$$

where $\mathbf{W}_k = [w_{1k}, w_{2k}, \ldots, w_{Nk}]^T$ represent weight vector and $\psi(X_k) = [\phi_1(X_k), \phi_2(X_k), \ldots, \phi_N(X_k)]^T$ is trigonometric expansion of the input vector. This output is then passed through a nonlinear function (a sigmoid function) to produce the estimated output

$$\rho(S_k) = \frac{1 - exp^{-S_k}}{1 + exp^{-S_k}} \tag{5.14}$$

The error signal e_k is the difference between the desired response and the model output and is given by

$$e_k = (d_k - y_k) \tag{5.15}$$

The error e_k and the input vector X_k are employed to the weight update algorithm to compute the correction weight vector ΔW_k. Let the reflected error is given by

$$\delta_k = e_k\ \rho(S_k)' = e_k\ \delta_k$$

where δ_k represents the derivative of the activation function given in (5.14). Then the updated weight vector is given by

$$\Delta_k = \delta_k\ \ (\psi(X_k))^T$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu\Delta_k + \gamma\Delta_{k-1} \tag{5.16}$$

where $\delta_k = (1 - y_k^2)e_k$. In these equations μ is the learning factor and γ is the momentum factor that helps to accelerate the speed of convergence of algorithms. When the training process is complete, the connecting weights of the model are frozen to their final values. The model so developed is then used for testing with known input codes, used for pulse compression.

5.4.3 Simulation Studies

The performance of the approach was studied using the FLANN structure shown in Fig. ??. The input patterns to the FLANN used for training it were the time-shifted sequences of the three codes, 13 element barker code, 21 element optimal code and 35 element combined barker code, assuming there are strings of zeros on either side of the given input code. This makes $(2 * N - 1)$ input patterns, including a null sequence where N is the length of the given Barker code. The desired output when the correct Barker code at the input is 1, and 0 when the input is any other time shifted sequence. But using only these $(2 * N - 1)$ input patterns made convergence very difficult, as there was only one input pattern among these input patterns whose desired output was 1; the rest being 0. To overcome this problem, we trained this model for 500 iterations. The performance of the proposed network is compared with the direct auto correlation function and Multi-layer Perceptron (MLP) with 13-3-1 structure using back prorogation learning algorithm [5.12, 5.13].

Convergence performance

As shown in Fig. ??, the convergence speed of the BP algorithm is inherently slow. The proposed approach based on FLANN has much better convergence speed and very low training error compared to BP algorithm for the 13 element barker code.

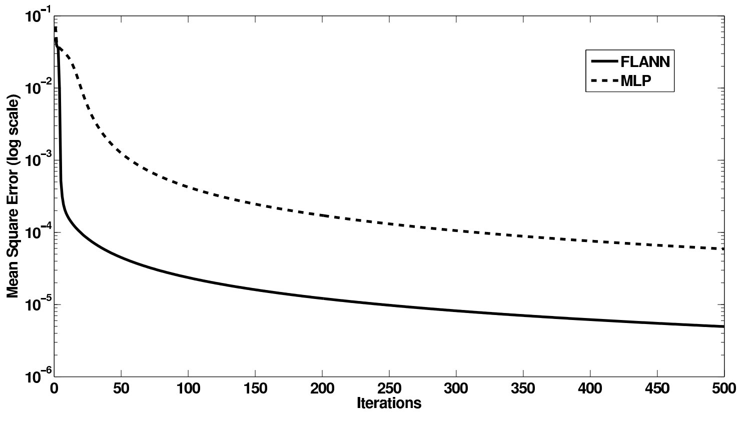
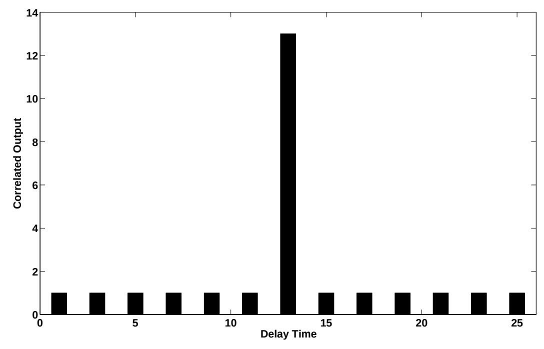


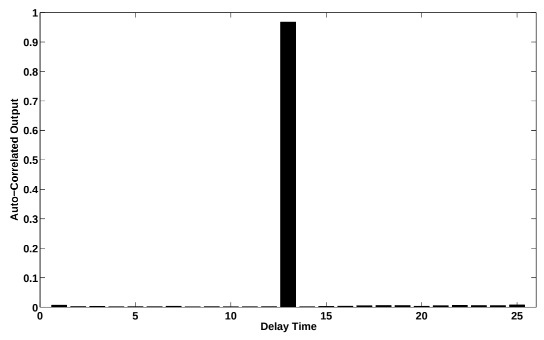
Figure 5.3: FLANN : Mean Square Error Plot

Signal-to-sidelobe ratio performance

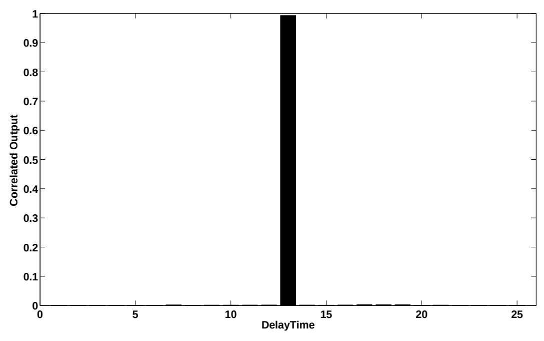
The SSR is defined as the ratio of peak signal amplitude to maximum sidelobe amplitude. Fig. ?? shows the compressed waveforms of 13-element Barker code using ACF, MLP and FLANN approach. The results of the investigation are depicted in Table ??. It shows that the proposed FLANN approach achieved, higher output SSR compared to other approaches in all the cases.



(a) Using ACF



(b) Using MLP



(c) Using FLANN

Figure 5.4: FLANN : Peak Signal to Side-lobe Ratio (SSR) for the 13 element barker code

Noise performance

In the radar target detection, the signal from the target is corrupted by noise, it is important to test the algorithm by adding noise to the pulse. The input signals are corrupted by white Gaussian noise with different signal to noise ratio (SNR). The performance of

Algorithms	SSR (dB)		
	13-element Barker code	21-element Optimal code	35-element Barker code
ACF	22.27	20.42	13.97
MLP	41.82	43.27	41.03
FLANN	49.59	52.41	50.79

Table 5.4: FLANN : Comparison of SSR (in dB)

the ACF, MLP and FLANN for the noisy case is shown in Tables **??** - **??**. From these tables, it is clear that the performance of FLANN is much better than any other approach accept for some SNR 3, 5 dB for 35 element combined Barker code. As shown in Fig. **??**, FLANN have better side lobe reduction than the other methods in 3 dB noisy condition.

Algorithms	SSR (dB)				
	SNR = 0 dB	3 dB	5 dB	10 dB	30 dB
ACF	4.42	9.71	12.65	17.07	21.69
MLP	17.11	35.67	39.27	42.48	41.91
FLANN	20.56	41.18	45.82	49.11	49.66

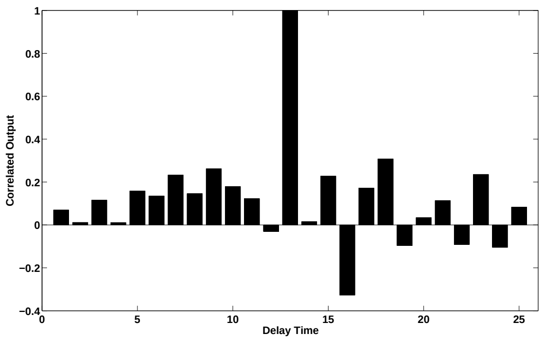
Table 5.5: FLANN : Comparison of SSR (in dB) for 13 element Barker code for different noisy condition

Algorithms	SSR (dB)				
	SNR = 0 dB	3 dB	5 dB	10 dB	30 dB
ACF	6.08	10.35	11.96	15.06	19.77
MLP	6.53	19.57	27.55	38.37	42.94
FLANN	26.11	36.41	41.19	48.19	52.75

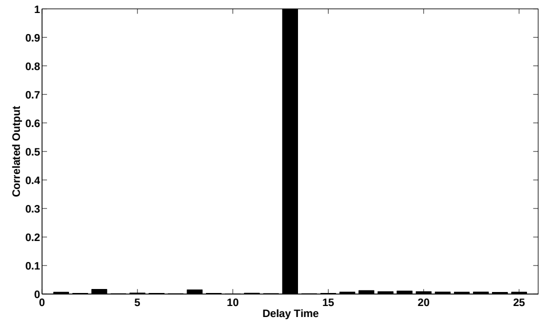
Table 5.6: FLANN : Comparison of SSR (in dB) for 21 element Optimal code for different noisy condition

Algorithms	SSR (dB)				
	SNR = 0 dB	3 dB	5 dB	10 dB	30 dB
ACF	7.98	9.81	10.88	12.21	13.78
MLP	22.91	35.17	40.58	43.40	41.30
FLANN	25.30	34.58	38.83	45.09	50.42

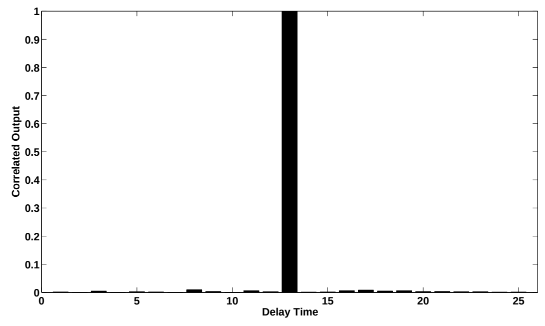
Table 5.7: FLANN : Comparison of SSR (in dB) for 35 element Combined Barker code for different noisy condition



(a) Using ACF



(b) Using MLP

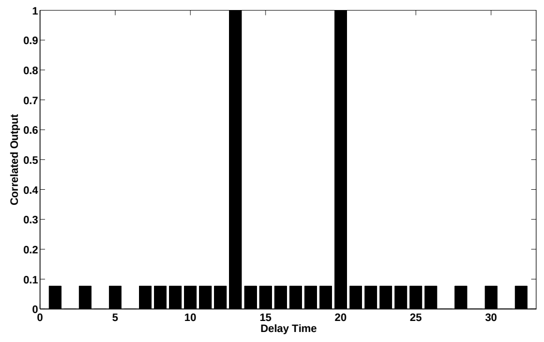


(c) Using FLANN

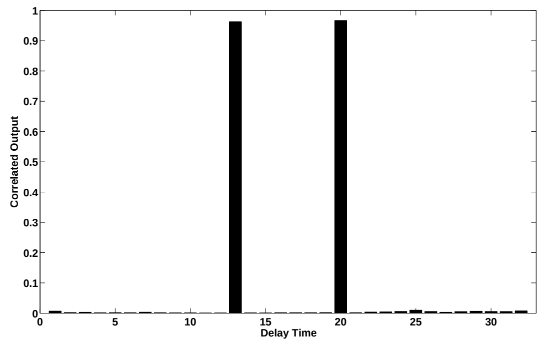
Figure 5.5: FLANN : Noise performance for the 13 element barker code for 3 dB SNR

Range resolution

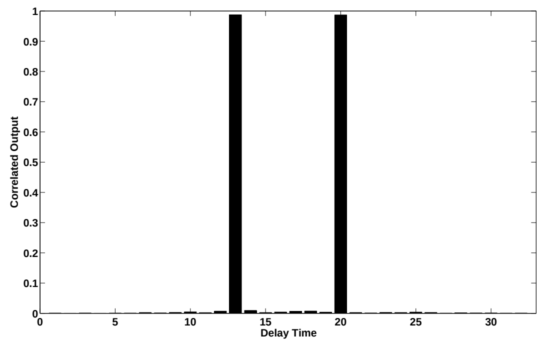
The range resolution is the ability to distinguish between two targets solely by measurements of their ranges in radar systems. The network should be able to distinguish between two close-by targets and should be able to resolve two overlapped targets. To make the comparison of the range resolution ability, we consider 13-element Barker code with two n-delay apart (DA) overlapping sequences having same magnitude ratios. The results in Tables **??** clearly indicate the superior performance of FALNN over other techniques accept at 5 delay. Fig. **??** show the examples of compressed waveforms of overlapped



(a) Using ACF



(b) Using MLP



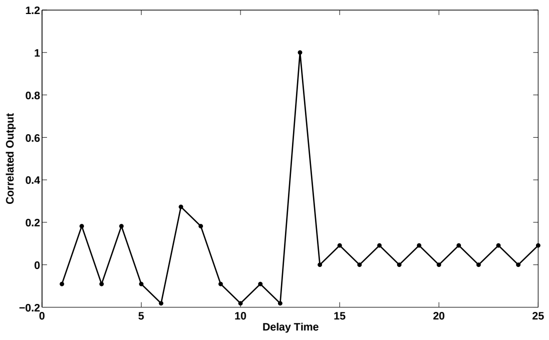
(c) Using FLANN

Figure 5.6: FLANN : Compressed waveforms of overlapped 13 element barker code with 7 delay apart

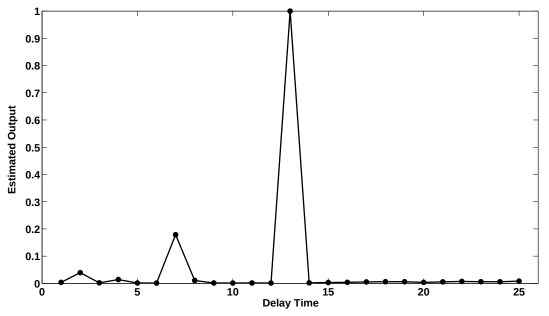
13-element Barker codes using ACF,MLP and FLANN approach.

Doppler shift performance

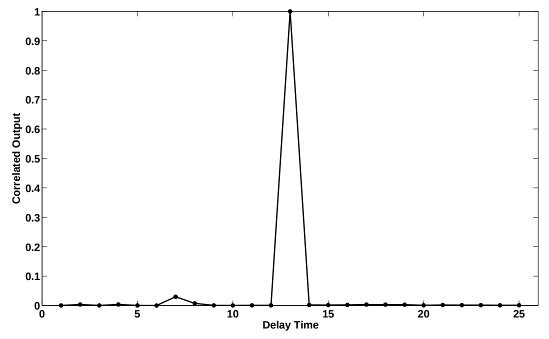
To check the Doppler tolerance of the pulse compression algorithms in this work, we consider 13-element Barker code. By shifting phase of the individual elements of the phase code, the Doppler sensitivity is caused. In the extreme, if the last element is



(a) Using ACF



(b) Using MLP



(c) Using FLANN

Figure 5.7: FLANN : Correlated output for Doppler Shift

Algorithms	SSR (dB)				
	1 delay	3 delay	5 delay	7 delay	15 delay
ACF	22.27	22.27	22.27	16.90	22.27
MLP	39.95	38.24	41.83	39.54	41.83
FLANN	40.19	39.93	38.70	40.23	43.39

Table 5.8: FLANN : Comparison of SSR (in dB) for 13 element Barker code for Range Resolution of two targets

shifted by 180^0, the code word is no longer matched with the replica. That is, the used sequence of pulse compression is changed from $\{1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1-1, 1\}$ to $\{-1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1-1, 1\}$. The results of comparison of all algorithms are shown in Fig. **??**. Results show that RBFN has significant advantage of robustness in Doppler shift interference. From Fig. **??**, it is observed that the MLP, and ACF are more sensitive to the Doppler shift produced by a moving target than the FLANN. Form Table **??**, we conclude that FLANN gives better Doppler Shift performance in comparison to other methods.

Algorithms	SSR (dB)		
	13-element Barker code	21-element Optimal code	35-element Barker code
ACF	11.28	13.53	13.46
MLP	14.95	38.91	41.05
FLANN	30.46	44.38	50.77

Table 5.9: FLANN : Comparison of SSR (in dB) for Doppler shift

5.4.4 Conclusion

The section proposed a new method to obtain high range resolution and Doppler tolerance for various codes using low complexity FLANN method. The results are compared with these obtained by the standard ACF and BP based methods. Extensive simulation study indicates that the new method is not only simple but also efficient than the BP and ACF methods.

Chapter 6

Pulse Compression using RBF Network A Multiobjective Approach

CHAPTER 6

PULSE COMPRESSION USING RBF NETWORK : A MULTIOBJECTIVE APPROACH

6.1 Introduction

Pulse Compression has important roles in a radar system to achieve long detection range and high range resolution. For the long range detection, the long pulse is transmitted. But to achieve the high range resolution, shorter pulses are employed. So that there are trade-off between the range resolution and long range detection for a given length of pulse. To solve the dilemma between these two, Pulse Compression technique is used in the pulse radar detection. In this technique the long pulses are transmitted which provides high energy to detect long distant target and before detection process, the echo pulses are shortened as short pulses, which gives high resolution.

Artificial neural networks are widely used as model structures to learn a nonlinear mapping based on the training data set in many kinds of application field due to their powerful nonlinear mapping ability [6.1]. The primary importance in applying neural networks to the Pulse Compression is in selecting its structure rather than the connection weights learning algorithms. Network structure is characterized by number of hidden layer, the number of hidden layer's unit and the response function. Several approaches to Pulse Compression using artificial neural networks have been proposed in last two decades, but a general method to determinate the structure of neural network has been established. This caused by the dilemma about the neural architecture complexity [6.2].

If a structure is complex, the generalization ability is low because of high variance error. Conversely, if a structure is simple then it can't represent the relation for input and output enough because of high error. So that design involves the optimization of two competing objectives, namely, the maximization of network capacity and minimization of neural architecture complexity. So in this chapter we show the approach to obtain the set of trade-off structures of RBF networks based on multiobjective optimization [6.3]. We use 13 element barker code, 21 bit optimal code and 35 element combination barker code for the training and testing of RBF networks. By this method, Pareto optimal solutions, using multi-objective algorithm, are obtained which are giving superior performance with less structure complexity compared to fixed center RBF network [6.2].

If the parameters of RBF networks i.e. the number of basis functions and the widths and centers of each basis function, are determined, output layer weights can be calculated with the training data with the help of pseudo inverse matrix (linear square method) [6.4]. The designers will be able to select one structure of RBF from the Pareto optimal structure set obtained by the proposed method according to their use or specific criteria.

6.2 Multiobjective Algorithm: Binary Nondominated Sorting Genetic Algorithm (BNSGA)

In this study BNSGA is used as the multiobjective computation [6.5]. BNSGA is well known as a multiobjective genetic algorithm which can maintain diversity on the Pareto front well. The chromosomes in this study are binary strings. The procedure of BNSGA is explained below:

At first initialize a population P_0 (a set of chromosomes by assigning binary strings). Then population P_0 is sorted based on nondomination in to front. The first front shows nondominant set in the current population and the second front being dominated by the individuals in the first front only and the front goes on. Each individual in each front are assigned rank (fitness) values. Individuals in the first front are given a rank 1 and so on. In addition to fitness value a new parameter called crowding distance is calculated for each individual. The crowding distance is a measure of how close an individual is to its neighbors. Large average crowding distance will result in better diversity in the population. Parents are selected from the population by using binary tournament selection, based on the rank and crowding distance. An individual is selected if the rank is lesser than the other. For individuals of same rank, individual with greater crowding distance have priority, and it will selected. The selected population generates offspring Q_0 from binary single point crossover and bit reversal mutation operators. The population

R_0 generated by merging the current population P_0 and current offspring Q_0, is sorted again based on nondomination and only best N individuals are selected, where N is the population size. This is repeated until the end condition is met.

6.3 Radial Basis Function (RBF) Network

A RBF network can be regarded as a special two layer network which is linear in the parameters by fixing all RBF centers and non-linearity in the hidden layer [6.4]. Thus the hidden layer performs a fixed non-linear transformation with no adjustable parameters and it maps the input space on to a new space. The output layer then implements a linear combiner on this new space and the only adjustable parameters are the weights of this linear combiner. These parameters can be therefore being determined using the linear least square method which is an important advantage of this approach. However the performance of an RBF network critically depends upon the chosen centers. In practice the centers are often chosen to be a subset of input data set. Typically the Gaussian function is used for the basis function in RBF network. It is defined as

$$\phi(||\mathbf{x}, \mathbf{c}_j||) = exp(-\frac{m}{d_{max}^2}||\mathbf{x} - \mathbf{c}_j||^2), \qquad j = 1, 2, \ldots, m \tag{6.1}$$

Here, d_{max} is the maximum distance between these selected centers and m is the number of centers. In effect, the standard deviation (i.e., width) of all the Gaussian radial basis functions is fixed at

$$\sigma = \frac{d_{max}}{\sqrt{2m}} \tag{6.2}$$

This formula ensures that the individual radial basis functions are not too peaked or too flat, both of these two extreme conditions should be avoided. As an alternative , we may use individually scaled centers with broader widths in areas of lower data density, which requires experimentation with the trained data.

A schematic of RBF network with M inputs and one output is depicted in Fig. **??**. Such a network implements a mapping $f_r(x) : \Re^M \to \Re$ according to

$$f_r(\mathbf{x}) = w_0 + \sum_{j=1}^{N} w_j \phi(||\mathbf{x} - \mathbf{c}_j||) \tag{6.3}$$

where $||.||$ denotes the Euclidean distance, w_j, $0 \le j \le N$ are output layer weights. By providing a set of the input $\mathbf{x}(t)$ and the corresponding desired output $d(t)$ for $t = 1$ to n, the values of the weights w_j can be determined using the linear LS method. A straight forward procedure for updating the weight vector, is to use the *pseudo inverse method* (Broomhead and Lowe 1988). Updating equation given by

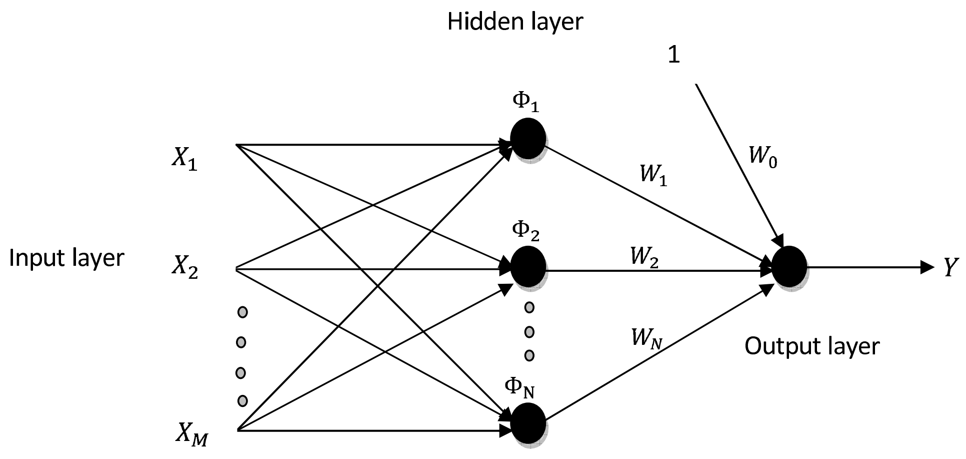


Figure 6.1: Radial Basis Functions (RBF) Network

$$\mathbf{w} = \phi^{+}\mathbf{d} \tag{6.4}$$

where **d** is the desired response vector in the training set. The matrix ϕ^{+} is the pseudo inverse of matrix ϕ; that is

$$\phi^{+} = (\phi^{T}\phi)^{-1}\phi^{T} \tag{6.5}$$

In practice, the centers are normally chosen from the data points $\{\mathbf{x}(t)\}_{t=1}^{n}$. The key question is therefore how to select center appropriately from the data set. In order to achieve a given performance, an unnecessarily large RBF network may be required. This adds computational complexity. Alternatively, the multi-objective algorithm can be select centers so that adequate and parsimonious RBF network can be obtained.

6.4 BNSGA based RBF Neural Network for Pulse Compression (Pareto RBF Network)

6.4.1 Genetic representation (Structure selection of RBF network)

BNSGA is used for the selection of centers for the RBF network. For the selection of network, initialize binary chromosome of BNSGA with the same length of the training data set. The chromosomes of BNSGA population indicate that each data point is employed as a center of basis functions or not. That is "1" in chromosome represents that a center of basis function is located at the corresponding training data point, as shown in Fig. **??**.

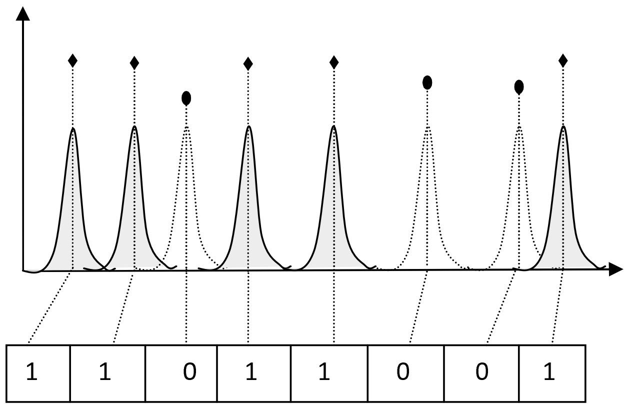


Figure 6.2: Chromosome - Center representation

In the chromosome, the position of gene value “1” indicate the center position of the basis function (selected center) and number of “1” genes in chromosome indicates the number of basis functions (number of centers). The training dataset is formed with the time shifting of barker code. For the length of N elements barker code, the number of input data set becomes $2N-1$. So for this we pad zeros to the barker code. For different input data point there are different outputs. So we take the length of chromosomes same as input dataset applied to RBF network.

6.4.2 Multiobjective fitness evaluation

The ANN design is cost as an MOO problem where a number of objectives such as training accuracy and degree of complexity are specified. The conflicting objectives of maximizing network capacity and minimizing network complexity is manifested in the trade-offs between training and test accuracy. The two objectives we consider in our multi-objective approach are first minimization of number of centers in the hidden layer (f_1), indicates the complexity of RBF network and second, the minimization of MSE (f_2), defined as

$$\begin{aligned} MSE &= \frac{1}{n}\sum_{i=1}^{n}(d_i - \widehat{d_i})^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(d_i - \left(\sum_{j=1}^{N} w_j \phi(||x_i - c_j||) + w_0\right)\right)^2 \end{aligned} \tag{6.6}$$

here d_i is desired output and $\widehat{d_i}$ estimated output during the training of RBF network. d_i is 1 when all elements of barker code present at the input nodes of RBF network, otherwise it become zero.

6.5 Performance Comparison

In this section, the proposed Pareto RBF network applied for the pulse compression in radar system and evaluated by numerical experiments. We dealt with the 13 element barker code, 21 bit optimal code and 35 element combine barker code. The performance of the proposed network is compared with the direct auto correlation function and fixed structure RBF network using supervised learning strategies where the number of hidden nodes fixed to 7 and take radius of attraction r=1 around the training patterns [6.6]. Parameters chosen for BNSGA is shown in Table **??**.

Population Size	100
Chromosome Type	Binary Number Representation
Selection	Binary Tournament Selection
Crossover Method	Single Point Binary Crossover
Crossover Probability	0.8
Mutation Method	Bit Reversal Mutation
Mutation Probability	0.2
Generation	200

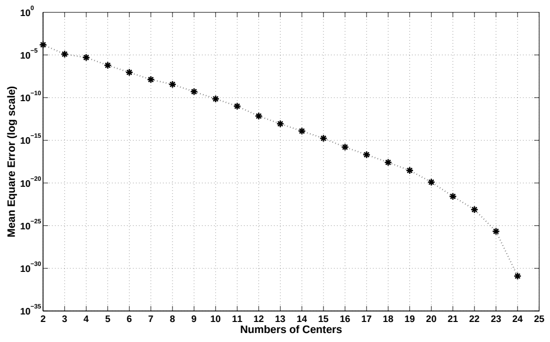
Table 6.1: Parameter Setting for BNSGA

6.5.1 Training of Pareto RBF network and Convergence Performance

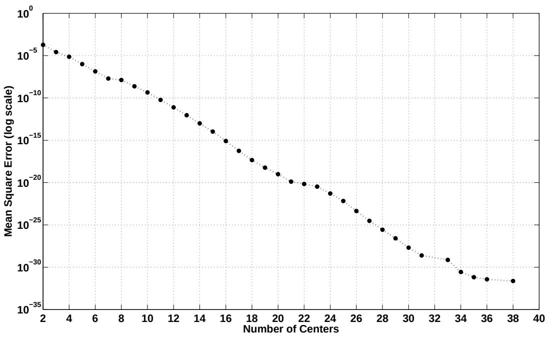
For the training of proposed network, we use 13 element barker code, 21 bit optimal code and 35 element combined barker code. We plot the Mean square error (MSE) vs Number of Centers plot for all codes in Fig. **??**. These are represent the Pareto Fronts. That shows, for increasing the number of centers, MSE going down, but at some number of centers, it may be increases, so that the multiobjective algorithm did not take these centers. By this unnecessary complexity of network is solved.

6.5.2 Testing of Pareto RBF network

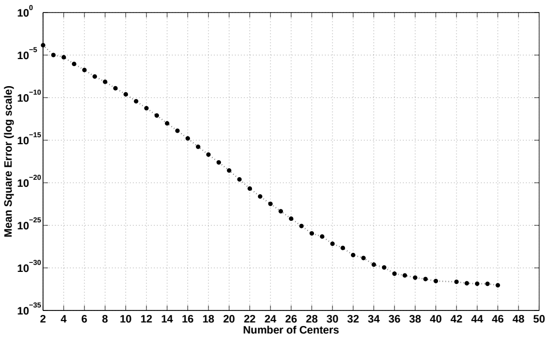
In this part, The Pareto RBF with 5-centers and 7-centers have taken for performance comparison. In the testing part, we use same signal of 13 element barker code, 21 element optimal code and 35 element combined barker code for performance comparison of peak signal to side-lobe ratio. There is the noise performance, to check the how much the given network is robust for different noisy condition. We find out the range resolution and Doppler shift ability for the given network.



(a) MSE vs Centers plot for 13 element Barker code



(b) MSE vs Centers plot for 21 element Optimal code



(c) MSE vs Centers plot for 35 element Combined Barker code

Figure 6.3: PRBF : Pareto Fronts

Signal to peak side-lobe ratio (SSR)

It is define as the ratio of the peak signal to the maximum side-lobe amplitude. In the testing part, SSR performance comparison results of the ACF, RBFN and Pareto RBF are shown in Table **??** for 13 element barker code, 21 element optimal code and 35 element combined barker code. From table, Pareto RBF with 5 centers given better SSR ratio than the fixed centers RBF network (with 7-centers), so from this we can say that, Pareto RBF have less structure complexity and gives better result than the other methods.

Algorithms	SSR (dB)		
	13-element Barker code	21-element Optimal code	35-element Barker code
ACF	22.27	20.42	13.97
RBFN (7-Centers)	64.90	60.18	71.70
PRBF (5-Centers)	85.53	126.25	196.87
PRBF (7-Centers)	94.23	132.11	213.09

Table 6.2: PRBF : Comparison of SSR (in dB)

Noise performance

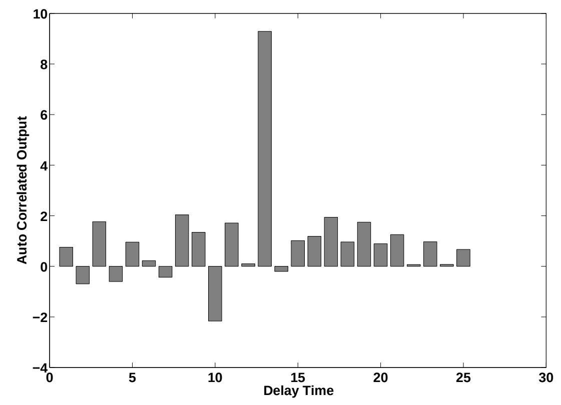
For different noise condition, we added the White Gaussian noise to the input signal with different signal to noise ratio (SNR). Table **??** - **??** shows the robustness of all three algorithms against the different noise environment. We take the five different $SNR = 2, 5, 10, 20, 30$ respectively. Pareto RBF given better side-lobes reduction in comparison two other method for all given different noisy condition. As shown in Fig. **??**, for the 13 element barker code with $SNR = 5$ dB, our proposed method is given better side-lobes reduction in comparison to other two method.

Algorithms	SSR (dB)				
	SNR = 2 dB	5 dB	10 dB	20 dB	30 dB
ACF	8.09	12.65	17.07	20.48	21.69
RBFN (7-Centers)	18.09	25.45	33.71	38.71	39.15
PRBF (5-Centers)	44.20	53.26	63.16	76.01	86.31
PRBF (7-Centers)	45.11	54.22	65.16	76.21	86.65

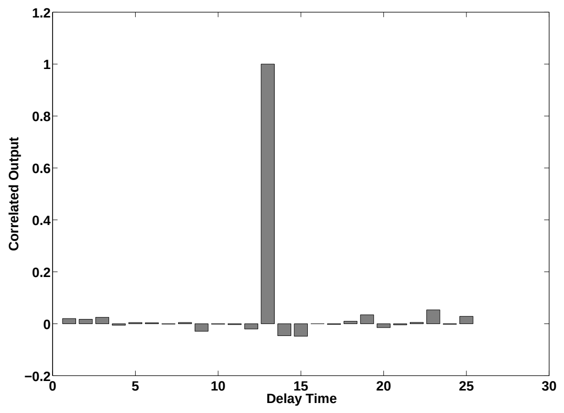
Table 6.3: PRBF : Comparison of SSR (in dB) for 13 element Barker code for different noisy condition

Algorithms	SSR (dB)				
	SNR = 2 dB	5 dB	10 dB	20 dB	30 dB
ACF	9.10	11.96	15.06	18.47	19.77
RBFN (7-Centers)	20.04	25.27	32.74	45.00	45.54
PRBF (5-Centers)	70.23	90.69	107.02	111.87	120.54
PRBF (7-Centers)	72.80	91.32	109.08	113.49	121.14

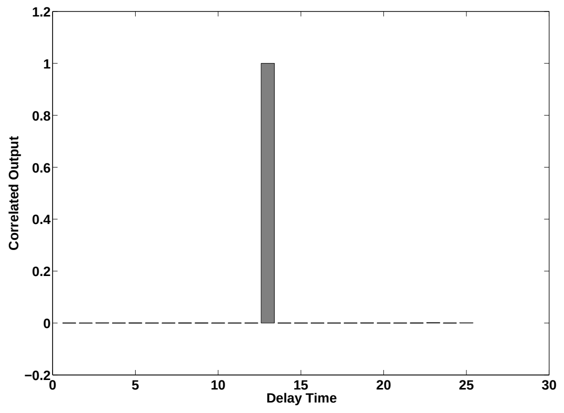
Table 6.4: PRBF : Comparison of SSR (in dB) for 21 element Optimal code for different noisy condition



(a) Using ACF



(b) Using RBFN



(c) Using ORBF

Figure 6.4: PRBF : Noise performance for the 13 element barker code for 5 dB SNR

Range Resolution ability

Range resolution is defined as the ability to separate two targets of similar reflectivity. Due to very nearly object, the echo pulses are overlapped. So to find out the ability of

Algorithms	SSR (dB)				
	SNR = 2 dB	5 dB	10 dB	20 dB	30 dB
ACF	8.50	10.66	12. 42	13.49	13.82
RBFN (7-Centers)	25.01	32.56	41.92	45.29	45.82
PRBF (5-Centers)	141.98	152.25	155.35	165.82	176.11
PRBF (7-Centers)	143.56	153.92	158.66	167.88	177.53

Table 6.5: PRBF : Comparison of SSR (in dB) for 35 element Combined Barker code for different noisy condition

range resolution, take mixture of two same pulses with Q delays apart as input to the network. The added input waveform of two equal magnitude 13-element barker codes of 4-delay apart is shown in Fig. **??**. The comparison of algorithms taken overlapped two 13-element barker code with same magnitude with different delays shown in Table **??**. The compressed wave form is shown in the Fig. **??**. The corresponding comparison with overlapped two 35-element combined barker code shown in Table **??**. The results in Tables **??** and **??** clearly indicate the superior performance of Pareto RBF over other techniques.

Algorithms	SSR (dB)				
	4 delay	5 delay	8 delay	10 delay	15 delay
ACF	16.90	22.27	16.90	16.90	22.27
RBFN (7-Centers)	27.25	24.67	31.36	30.63	37.14
PRBF (5-Centers)	42.25	46.31	64.92	74.99	92.65
PRBF (7-Centers)	44.04	48.05	67.42	78.14	94.23

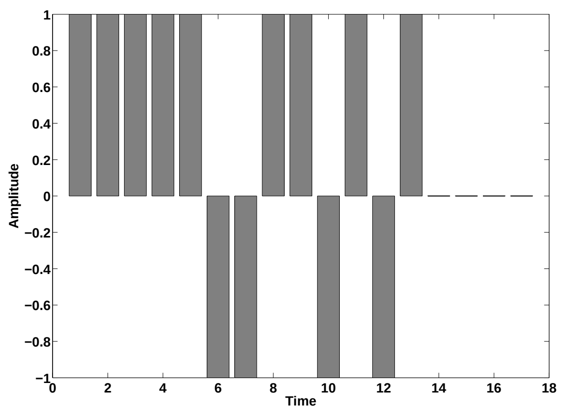
Table 6.6: PRBF : Comparison of SSR (in dB) for 13 element Barker code for Range Resolution of two targets

Algorithms	SSR (dB)				
	4 delay	5 delay	8 delay	10 delay	15 delay
ACF	9.54	13.97	10.62	10.62	13.97
RBFN (7-Centers)	25.69	28.36	29.51	31.59	35.14
PRBF (5-Centers)	45.20	43.89	64.99	77.56	103.75
PRBF (7-Centers)	46.88	45.74	68.33	79.05	105.86

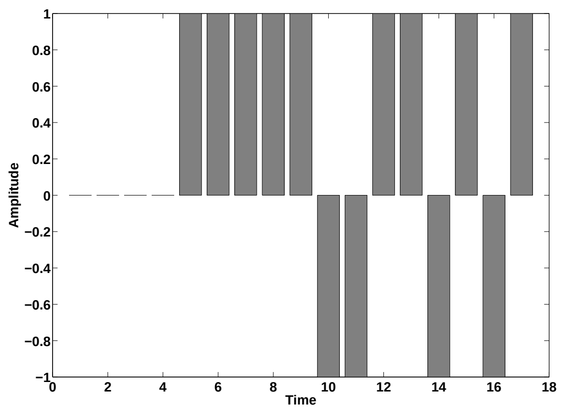
Table 6.7: PRBF : Comparison of SSR (in dB) for 35 element Combined Barker code for Range Resolution of two targets

Doppler Shift Performance

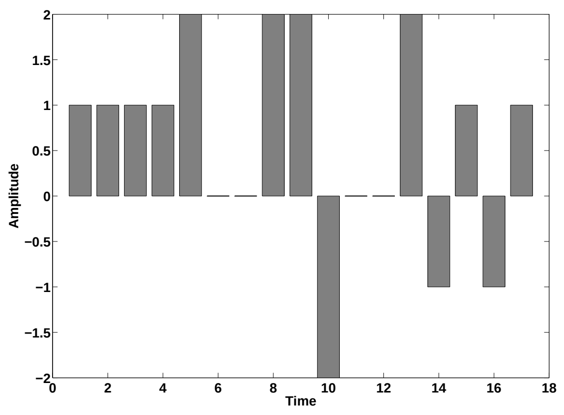
Due to the moving target, echoes returning from the target are processed to measure the frequency shift between carrier cycle in each pulse and the original transmitted frequency.



(a) Left shift



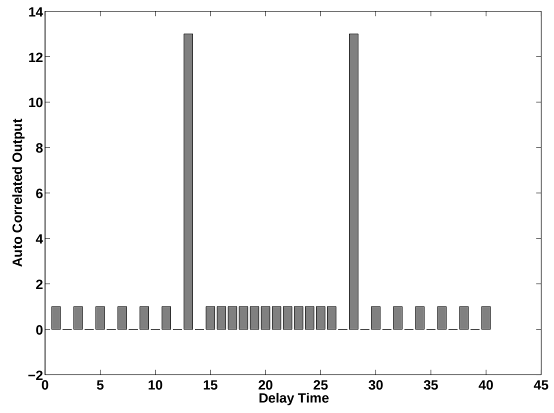
(b) Right shift



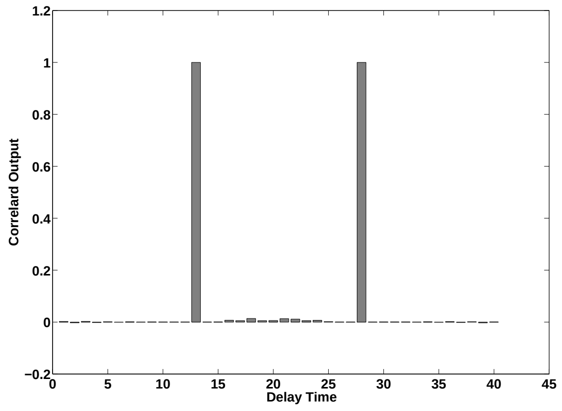
(c) Added input waveform

Figure 6.5: Input waveform on additional of two 4-delay-apart 13-element Barker sequence having same amplitude

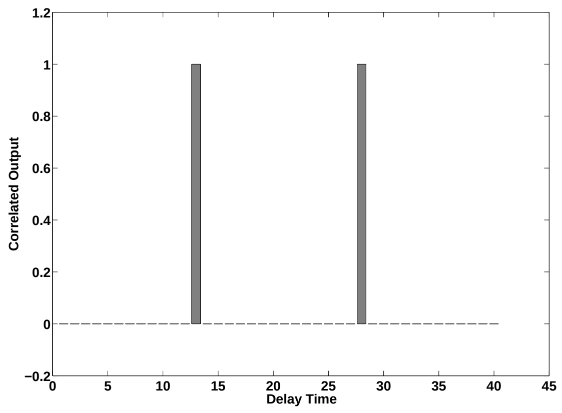
This implies the presence of Doppler- shifted signal. It can be accounted by the shifting in phase of individual elements of the phase code. To examine the Doppler tolerance



(a) Using ACF



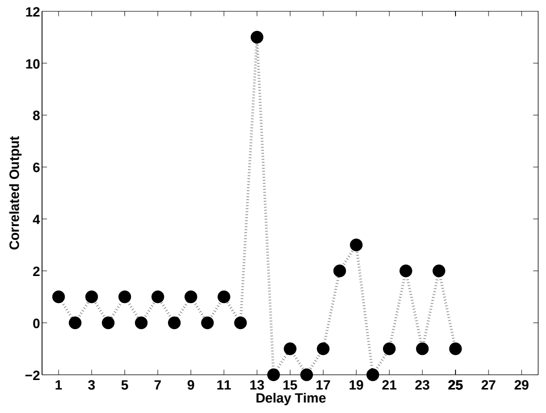
(b) Using RBFN



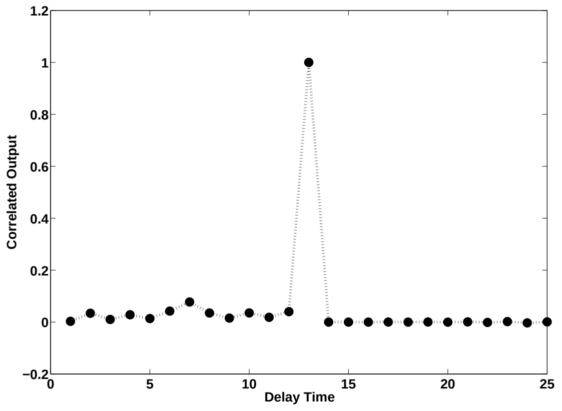
(c) Using PRBF

Figure 6.6: PRBF : Compressed waveforms of overlapped 13 element barker code with 15 delay apart

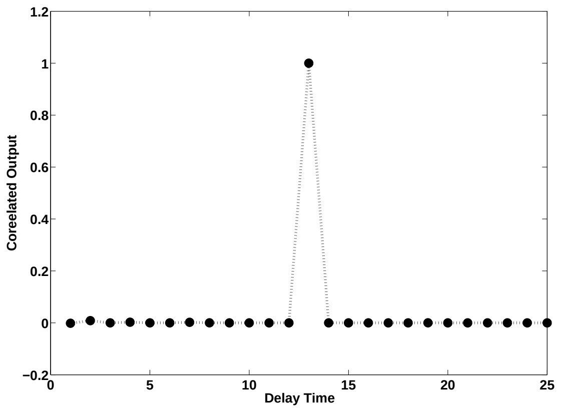
of Pulse Compression with different algorithms, taken extreme condition, in which the last element of code is shifted by180^o, so that the code word is no longer matched with



(a) Using ACF



(b) Using RBFN



(c) Using PRBF

Figure 6.7: PRBF : Doppler Shift Performance for 13 element barker code

the replica. That is, used 13- element barker code $\{1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1\}$ changed to $\{-1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1\}$. Comparison between all algorithms shown in Fig. **??** for 13-element barker code. The simulated result is shown in Table **??**

for all three codes. Results show that Pareto RBF has significant advantage of robustness in Doppler shift interference. From Fig. ??, it is observed that the ACF and RBFN are sensitive to the Doppler shift produced by a moving target.

Algorithms	SSR (dB)		
	13-element Barker code	21-element Optimal code	35-element Barker code
ACF	11.28	13.53	13.46
RBFN (7-Centers)	21.24	25.22	41.90
PRBF (5-Centers)	41.27	84.21	167.09
PRBF (7-Centers)	41.28	84.19	167.01

Table 6.8: PRBF : Comparison of SSR (in dB) for Doppler shift

6.6 Conclusion

A method of structure selection of RBF network from a viewpoint of multiobjective optimization for the pulse compression has been presented and extensive simulation have been carried out investigate the performance of the approach. By applying the proposed algorithm, wide variety of RBF network structures, in the sense of the Pareto optimum solution, are obtained and the availability of the proposed approach is shown by numerical simulation results. This new approach has a much superior output signal to sidelobe ratio than those of the traditional approaches such as the conventional ACF approach and the inverse filters and the fixed center RBF network. It has better range resolution and robustness in Doppler shift interference. This approach has higher SSRs in different adverse situations of noise.

Chapter 7

Conclusion and Scope for Future Work

CHAPTER 7

CONCLUSION AND SCOPE FOR FUTURE WORK

In this thesis, we have presented a new multiobjective optimization algorithm based on the bacteria foraging strategy. In contrast to most other MOO algorithms, for reproduction MOBFO select solutions with a probability that is dependent on the amount of domination measured in terms of the crowding distance between the solutions in the objective space. The approach is able to produce results similar or better than those generated by other two algorithms that are representative of the state of the art in evolutionary multiobjective optimization. The approach proposed uses a very simple mechanism to optimize test functions and our results indicate that such mechanism, despite its simplicity, is effective in practice. Therefore, we can conclude that bacteria foregoing optimization (BFO) can be effectively used to solve multiobjective optimization problems in a relatively simple way.

There are several ways in which the proposed MOBFO algorithm may be extended in future. In the MOBFO, the main part is step size, in future we can use adaptive step size in chamotactic loop for foraging purpose. We also believe that, given the features of foraging strategy, an extension of this paradigm for multiobjective optimization may be particularly useful to deal with high dimensional test functions and that is precisely part of our future research. We are also interested in analyzing alternative mechanism to maintain diversity and to achieve better distribution of solutions along the Pareto front, which is a current weakness of the approach proposed in this thesis. We are also considering the possibility of using spatial data structures to store and retrieve nondominated solutions in a more efficient way.

We have introduced a novel FLANN based structure for pulse compression in radar signal processing. Because of its single layer structure, the FLANN offer advantages over MLP as less computational complexity. The performance of the FLANN is found to be the best in terms of MSE level, convergence rate and signal to sidelobe ratio (SSR) for sidelobes reduction over a wide range of SNR. FLANN structure is also less sensitive to Doppler shift in comparison to other methods. Because of computational advantages the FLANN may be used in other signal processing applications.

In another chapter, a multiobjective evolutionary approach to ANN design is proposed. In this study, we have shown a method of obtaining a Pareto optimal RBF network based on multiobjective evolutionary algorithm. Then we have constructed an ensemble network by the Pareto optimal RBF networks applying them to pulse compression, and a performance of the ensemble network as a nonlinear system model has also been considered. MOO approach used for facilitating the exchange of neuronal information between candidate neural network designs and adaptation of the number of basis functions for each individual, based on a geometrical approach in identifying hidden-layer basis functions to prune. BNSGA efficiently obtain a set of Pareto solutions from initial solutions, which can be used for selecting number of centers of RBF network for pulse compression problem with few interactions among parameters. Numerical simulation results indicate that the ensemble network has an ability to reduce MSE with number of basis functions in RBF network. While we have demonstrated the effectiveness of our proposed approach for pulse compression, we believe that the methods , we have employed in this thesis are sufficiently flexible and robust to be extended to handle complexity and accuracy of RBF network. The proposed method has applicability to sidelobes reduction of received signal.

To find the users preference for tradeoff and appropriate settings of the ensemble weights are the further issues. These may depend on the application field or the purpose of pulse compression. Reduction of the computational cost, improvement of the ensemble technique and comparison to the conventional approaches are also the future works and they are under investigation.

References

Chapter-1

1.1 Rosenberg,R. S.(1967). *Simulation of Genetic Populations with Biochemical Properties.* Ph.D. Thesis, Ann Arbor, MI, University of Michigan.

1.2 Schaffer, J. D. (1984). *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms.* Ph. D. Thesis, Nashville, TN: Vanderbilt University.

1.3 Nathanson, F.E. (1969). *Radar Design Principles.* New York: McGraw-HIll, 1969, 452-469

1.4 Skolnik, M.I. *Introduction to radar systems.* McGraw-Hill Book Company Inc. 1962

1.5 Ackroyd, M.H., and Ghani, F. *Optimum mismatch filters for sidelobe suppression.* IEEE Trans. Aerosp. Electron. Syst., 1973, 9,(2), pp. 214218.

1.6 Mese, E., and Giuli, D. *Optimal recursive phase-coded waveforms radars.* IEEE Trans. Aerosp. Electron. Syst., 1977, 13, (2), pp. 163171

1.7 Kwan, H.K., and Lee, C.K.: *A neural network approach to pulse radar detection.* IEEE Trans. Aerosp. Electron. Syst., 1993, 29, (1), pp. 921

1.8 Deergha, R., and Sridhar, G.: *Improving performance in pulse radar detection using neural networks.* IEEE Trans. Aerosp. Electron. Syst., 1995, 31, (3), pp. 11931198

1.9 Chen, S.C., and Grant, P.M.: *Orthogonal least squares learning algorithm for radial basis function networks.* IEEE Trans. Neural Networks, 1991, 2, (2), pp. 302309

1.10 Duh, F.B., Juang, C.F., and Lin, C.T.: *A neural fuzzy network approach to radar pulse compression.* IEEE Geosci. Remote Sens. Lett., 2004, 1, (1), pp. 1520

1.11 D.G. Khairnar, S.N. Merchant and U.B. Desai : *Radial basis function neural network for pulse radar detection.* IET Radar Sonar Navig., 2007, 1, (1), pp. 817

1.12 Deb, K. (1999). *Multi-objective Genetic Algorithms: Problem difficulties and construction of test problems.* Evolutionary Computation Journal 7(3), 205-230

1.13 Deb, K.,Agrawal, S.,Pratap, A. and Meyarivan, T. (2000). *A fast and elitist non-dominated sorting genetic algorithms for multi-objective optimization: NSGA -II.* IEEE Trans. on Evol. Comp., VOL. 6, NO. 2, April 2002

1.14 S. Haykin, *Neural Networks: A comprehensive foundation.* 2nd Edition, Pearson Education Asia, 2002.

1.15 M. T.Hagan, H.B.Demuth, M.Beale, *Neural Network Design.* Thomson Asia Pte. Ltd ,Singapore,2002

1.16 Dayhoff E.J., *Neural Network Architecture An Introduction.* Van Norstand Reilold, New York, 1990.

1.17 Bose N.K., and Liang P., *Neural Network Fundamentals with Graphs, Algorithms, Applications.* TMH Publishing Company Ltd, 1998.

1.18 Lippmann, R.P.: *An introduction to computing with neural nets.* Ieee Assp Mag., 1987, 4, (2), pp. 422

1.19 D. S. Broomhead and D. Lowe. *Multivariable functional interpolation and adaptive networks.* Complex Systems, 2:321355, 1988.

1.20 J. Moody and C. J Darken. *Fast learning in networks of locally-tuned processing units.* Neural Computation, 1(2):281294, 1989.

1.21 Nobuhiko Kondo, Toshiharu Hatanaka and Katsuji Uosaki. *Structure Selection of RBF Network Using Multi-Objective Genetic Algorithm.* SICE Annual Conference in Fukui, August 4-6,2003.Fukui University, Japan

1.22 E. G. M. de Lacerda 2003, *Model Selection of RBF Networks Via Genetic Algorithms.*, Ph. D, Informatics Center of Pernambuco Federal University,Brazil March, 2003

Chapter-2

2.1 Deb, K., Pratap. A, Agarwal, S., and Meyarivan, T. (2002). *A fast and elitist multiobjective genetic algorithm: NSGA-II.* IEEE Transaction on Evolutionary Computation, 6(2), 181-197

2.2 Deb. K. *An efficient constraint-handling method for genetic algorithms.* Comput. Methods Appl. Mech. Eng., vol. 186, no. 2-4, pp. 311-338, 2000.

2.3 J. Horn, N. Nafploitis, and D. E. Goldberg, *A niched Pareto genetic algorithm for multiobjective optimization.* in Proceedings of the First IEEE Conference on Evolutionary Computation, Z. Michalewicz, Ed. Piscataway, NJ: IEEE Press, 1994, pp. 82-87.

2.4 J. Knowles and D. Corne, *The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization.* in Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ: IEEE Press, 1999, pp. 98-105.

2.5 Carlos A. Coello Coello, Gary B. Lamont. *Applications of multi-objective evolutionary algorithms.* Advances in natural computation, vol-1, Dec 2004

2.6 Deb. K. *Optimization for Engineering Design: Algorithms and Examples.* New Delhi: Prentice-Hall, Feb 2004.

2.7 Deb. K. *Multi-objective Optimization using Evolutionary Algorithms.* Chichester, U.K. Wiley, 2001

2.8 Carlos A. Coello Coello, Gary B. Lamont, David A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems.*, 2nd ed., 2007.

Chapter-3

3.1 Deb. K. *Multi-objective Optimization using Evolutionary Algorithms.* Chichester, U.K. Wiley, 2001

3.2 N. Srinivas and K. Deb, *Multiobjective function optimization using nondominated sorting genetic algorithms*, Evol. Comput., vol. 2, no. 3, pp. 221-248, Fall 1995.

3.3 Deb.K.,*An efficient constraint-handling method for genetic algorithms*, Comput. Methods Appl. Mech. Eng., vol. 186, no. 2-4, pp. 311-338, 2000.

3.4 Deb, K., Pratap. A, Agarwal, S., and Meyarivan, T. (2002). *A fast and elitist multi-objective genetic algorithm: NSGA-II.* IEEE Transaction on Evolutionary Computation, 6(2), 181-197.

3.5 K. Deb and R. B. Agrawal, *Simulated binary crossover for continuous search space*, in Complex Syst., Apr. 1995, vol. 9, pp. 115-148.

3.6 T. Ray, K. Tai, and C. Seow, *An evolutionary algorithm for multiobjective optimization*, Eng. Optim., vol. 33, no. 3, pp. 399-424, 2001.

3.7 Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga, *Handling multiple objectives with particle swarm optimization*,Evol. Comput., vol. 8, pp. 256-279, NO. 3, June 2004

3.8 Kennedy and R. C. Eberhart, *Swarm Intelligence.* San Mateo, CA: Morgan Kaufmann, 2001.

3.8 J. Knowles and D. Corne, *The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization,* in Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ: IEEE Press, 1999, pp. 98-105.

3.9 Deb. K, *Multiobjective genetic algorithms: Problem difficulties and construction of test functions*, in Evol. Comput., 1999, vol. 7, pp. 205-230.

3.10 A. Coello Coello and M. S. Lechuga, *MOPSO: A proposal for multiple objective particle swarm optimization*, in Proc. Congr. Evolutionary Computation (CEC'2002), vol. 1, Honolulu, HI, May 2002, pp. 1051-1056.

3.11 J. D. Knowles and D.W. Corne, *Approximating the nondominated front using the Pareto archived evolution strategy,* Evol. Comput., vol. 8, pp. 149-172, 2000.

3.12 K.M.Passino, *Biomimicry of Bacterial Foraging for distributed optimization and control,* IEEE Control Syst. Mag., vol.22, no.3, pp52-67, Jun.2002.

3.13 V. Gazi, K.M. Passino, Stability analysis of social foraging swarms, IEEE Transactions on Systems Man and Cybernetics Part B Cybernetics 34 (1) (2004) 539557.

3.14 Sambarta Dasgupta, Arijit Biswas, Ajith Abraham and Swagatam Das, *Adaptive Computational Chemotaxis in Bacterial Foraging Algorithm*, International Conference on Complex, Intelligent and Software Intensive Systems, 64-71, 2008

Chapter-4

4.1 Carlos A. Coello Coello, Gary B. Lamont. *Applications of multi-objective evolutionary algorithms.* Advances in natural computation, vol-1.

4.2 Deb. K. *Multi-objective Optimization using Evolutionary Algorithms.* Chichester, U.K. Wiley, 2001

4.3 Deb, K., Pratap. A, Agarwal, S., and Meyarivan, T. (2002). *A fast and elitist multi-objective genetic algorithm: NSGA-II.* IEEE Transaction on Evolutionary Computation, 6(2), 181-197.

4.4 Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga, *Handling multiple objectives with particle swarm optimization*,Evol. Comput., vol. 8, pp. 256-279, NO. 3, june 2004

4.5 J. D. Schaffer, *Multiple objective optimization with vector evaluated genetic algorithms,* in Proceedings of the First International Conference on Genetic Algorithms, J. J. Grefenstette, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 93100.

4.6 F. Kursawe, *A variant of evolution strategies for vector optimization,* in Parallel Problem Solving from Nature, H.-P. Schwefel and R. Mnner, Eds. Berlin, Germany: Springer-Verlag, 1990, pp. 193197.

4.7 K. Deb, *Multi-objective genetic algorithms: problem difficulties and construction of test problems,* Evol. Comput., vol. 7, pp. 205230, Fall 1999.

Chapter-5

5.1 Nathanson, F.E. (1969). *Radar Design Principles.* New York: McGraw-HIll, 1969, 452-469

5.2 Skolnik, M.I. *Introduction to radar systems.* McGraw-Hill Book Company Inc. 1962

5.3 Ackroyd, M.H., and Ghani, F. *Optimum mismatch filters for sidelobe suppression.* IEEE Trans. Aerosp. Electron. Syst., 1973, 9,(2), pp. 214218.

5.4 Mese, E., and Giuli, D. *Optimal recursive phase-coded waveforms radars.* IEEE Trans. Aerosp. Electron. Syst., 1977, 13, (2), pp. 163171

5.5 Kennedy and R. C. Eberhart, *Swarm Intelligence.* San Mateo, CA: Morgan Kaufmann, 2001.

5.6 Nunn. C. *Constrained optimization applied to pulse compression codes and filters.* IEEE in Radar Conf. 9-12 May 2005 pp 190-194

5.7 Richard O. Duda, Peter E. Hart and David G. Stork, *Pattern Classification*, 2nd edition, John Wiley and Sons, INC.2001.

5.8 Y.H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison Wesley, Reading, Massachusetts, 1989.

5.9 Patra J.C., and Pal R.N., *A Functional Link Artificial Neural Network for Adaptive Channel Equalization,* Signal Processing 43(1995) 181-195, vol.43, no.2, May 1995.

5.10 S. Haykin, *Neural Networks: A comprehensive foundation.* 2nd Edition, Pearson Education Asia, 2002.

5.11 B. Widrow and S. D. Stearns , *Adaptive Signal Processing*, Pearson Education, 2002.

5.12 Kwan, H.K., and Lee, C.K.: *A neural network approach to pulse radar detection.* IEEE Trans. Aerosp. Electron. Syst., 1993, 29, (1), pp. 921

5.13 Deergha, R., and Sridhar, G.: *Improving performance in pulse radar detection using neural networks.* IEEE Trans. Aerosp. Electron. Syst., 1995, 31, (3), pp. 11931198

Chapter-6

6.1 T. Hatanaka, K. Uosaki and T. Kuroda, *Structure Selection of RBF Neural Network Using Information Criteria*, Proceedings of Fifth International Conference on Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies, pp.166-170, 2001.

6.2 T. Hatanaka, N. Kondo and K. Uosaki : *Multi-objective structure selection for radial basis function networks based on genetic algorithm,* Proceedings of 2003 Congress on Evolutionary Computation, pp. 1095- 1100, 2003.

6.3 K. Deb , *Multi-objective Optimization using Evolutionary Algorithms*,John Wiley and Sons, New York, 2001

6.4 S. Haykin, *'Neural Networks': A Comprehensive Foundation*, second. edition, Prentice-Hall, 1999.

6.5 Deb, K., Pratap. A, Agarwal, S., and Meyarivan, T. (2002). *A fast and elitist multi-objective genetic algorithm: NSGA-II.* IEEE Transaction on Evolutionary Computation, 6(2), 181-197.

6.6 D.G. Khairnar, S.N. Merchant and U.B. Desai, *Radial basis function neural network for pulse radar detection*, IET Radar Sonar Navig., 2007, 1, (1), pp. 8-17