PERFORMANCE IMPROVEMENT OF GENETIC ALGORITHM USED FOR MULTIOBJECTIVE OPTIMIZATION

A Thesis submitted in the fulfillment of requirement for the award of

Degree of

Doctor of Philosophy

In Computer Science & Engineering

Under Faculty of Engineering & Technology

Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur

Submitted by

Rahila Begum A.R. Patel

Supervisor

Dr. M. M. Raghuwanshi

February 2014

Department of Computer Science & Engineering

G.H.Raisoni College of Engineering Nagpur

Nagpur 440016, M.S. (INDIA)

CERTIFICATE

This is to certify that, I have been supervising the work of

*Ms. Rahila Begum A.R. Patel on “***PERFORMANCE IMPROVEMENT OF GENETIC ALGORITHM USED FOR MULTIOBJECTIVE OPTIMIZATION***” for the degree of Doctor of Philosophy in Computer Science & Engineering under Faculty of Engineering and Technology. The work is comprehensive, complete and fit for evaluation. The contents of the thesis in part or whole have not been submitted to any other Institute/ University for the award of any degree/diploma.*

Supervisor

February 2014

Dr. M.M. Raghuwanshi

Principal
Rajiv Gandhi College of
Engineering & Research,
Nagpur

DECLARATION

I, hereby declare that the investigation presented in the thesis has been carried out by me in the Department of Computer Science & Engineering of G.H. Raisoni College of Engineering , Nagpur. The work is original and has not been submitted earlier as a whole or in part for a degree/diploma at this or any other Institute/ University.

February 2014 **Rahila Begum A. R. Patel**

G. H. Raisoni College of Engineering, Nagpur
(an autonomous Institute under UGC act 1956)

Certificate

This is to certify that **Ms. Rahila Begum A. R. Patel** has presented her pre-submission seminar before the committee and the thesis is approved and forwarded to Research and Recognition Committee of Computer Science & Engineering in the Faculty of Engineering & Technology, Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur

Supervisor

Dr. M. M. Raghuwanshi
Principal
Rajiv Gandhi College of
Engineering &
Research, Nagpur

Forwarded by –

Head
Department of Computer Science & Engineering
G. H. Raisoni College of Engineering, Nagpur

Dean R&D
G. H. Raisoni College of Engineering, Nagpur

Dr. P. R. Bajaj
Director
G. H. Raisoni
College of

Engineering
Nagpur

Date :

ACKNOWLEDGEMENTS

First and foremost, I feel indebted to my supervisor, **Dr M. M. Raghuwanshi, Principal,** Rajiv Gandhi College of Engineering, Nagpur, for his valuable guidance, continuous support, advice and constant encouragement throughout my research work. I am proud to be his student and grateful to him for very capably guiding me through this endeavor and especially for coming out with wonderfully insightful solutions in moments of crisis & confusion. This work would have been impossible without his feedback, patience and kindness. It is not only a pleasure but also an honor and privilege to acknowledge the profound philosophical & conceptual influence that Dr. P. N. Suganthan, Nanyang Technological University, Singapore, has had on my research work. In particular, I would specially like to thank him for guidance and support

I would like to extend my gratitude to honorable **Dr. P. R. Bajaj**, Director, G. H. R. C. E., Nagpur, for being a constant source of inspiration. I am also grateful to Dr. S.B. Jaju, Dean R&D, G. H. R. C. E, Nagpur, and Dr. L.G. Malik, Head, Dept. of Computer Science and Engg., G. H. R. C. E, for allowing me use institutional and departmental resources very generously for carrying out my research. I would like to thank Prof. Veena Gulhane, R&D coordinator and other faculty members of Department of CSE, G. H. R. C. E, Nagpur for their help and support.

I would like to thank **Shri. S.R. Potdukhe, President** SPMS trust and **Dr. K.R. Dixit, Principal,** RCERT for giving me all the support and facilities to do research activity. Thanks are due to many people who helped me during this research work especially my friends and fellow departmental colleagues, because they deserve it and are not thanked nearly enough.

I shall be, forever, grateful to my mother Mrs. Hamida A.R. Patel for her encouragement, support and blessings through the years. I must thank to my

Husband Mr. Hamid Sheikh and my kid *Wasif* for their love and support that provided me strength, self-confidence and reliance during the entire period. I would like to take this opportunity to thank my sister, Brother-in-law and their kids Ameen & Nargis. I am thank full to my brothers and their family for all the help.

I dedicate this work to my **father Late Mr. Abdul Rakib A. G. Patel.**

Finally, my thanks to one and all, who have, in any way, contributed in accomplishing this work.

Rahila Patel

Index

LIST OF ABBREVATIONS

MOP	Multi-objective Optimization Problems
OR	Operations Research
SOP	Single Objective Optimization Problem
MOEAs	Multi-Objective Evolutionary Algorithms
GA	Genetic Algorithm
EA	Evolutionary Algorithm
MOGA	Multi-Objective Genetic Algorithm
RCGA	Real-Coded Genetic Algorithm
PCCO	Parent Centric Crossover Operator
BLX-α	blend crossover with α
SBX	Simulated Binary Crossover
SPX	simplex crossover
UNDX	unimodal normal distributed crossover
PCX	parent-centric crossover
SBX-l	SBX operator with lognormal probability distribution
G3	Generative Generation Gap
NSGA	Non-dominated Sorting Genetic Algorithm
MPX	Multi-parent Polynomial distribution crossover
MLX	Multi-parent LogNormal distribution crossover

SPC	Scaled Probabilistic Crowding
PNX	Parent Centric Normal crossover
CMA-ES	Evolutionary Strategy with Covariance Matrix Adaptation
MMX	Multi-parent mix probability distribution recombination
MPLX	Multi-parent Polynomial and Lognormal distribution based recombination
MHX	Multi-Parent Hybrid Recombination
NAM	Negative Assortative Mating
UFS	Uniform Fertility Selection
PBX- α	Parent-Centric BLX-α Crossover
OBL	Opposition Based Learning
EDS-MOGA	Ensemble of Dying Strategy based MOGA

LIST OF FIGURE

LIST OF TABLES

Chapter 1

Multi-objective Optimization and Genetic Algorithm

This chapter aims to provide background knowledge Multi-objective optimization problems (MOP) and genetic algorithm. Chapter begins with the discussion on Multi-objective optimization basic concepts, requirements of optimizer and MOP test suit. Evolutionary Algorithms (EAs) are based on some of the phenomenon of the nature. Genetic Algorithms (GAs) are evolutionary algorithms. This chapter introduces working principles of simple GA and discusses specific issues like population initialization, selection mechanism, crossover operator, replacement strategies and GA parameters. This chapter also gives insight to EAs suitability for solving MOP and covers literature review as well as metrics for performance measure of EA for MOP.

Multi-objective Optimization

Problem solving is a cognitive process and one of the most complicated intellectual activity of the human brain. Regardless of the problem's nature, the process of problem solving is concerned with finding solutions to certain predicaments or transitions from certain reviled states to desired states, such as the transition from unstable to stable or incorrect to correct. When a problem exists, a solution is sought and an objective is established. Unfortunately, finding a solution to certain problems or satisfying certain objectives can be a tricky and complicated process. An exact solution to some problems might simply be infeasible, especially if the problem consists of multiple conflicting and constrained objectives. In many applications, a good approximation or alternatively an *optimized* estimate to a solution might be deemed very good and sufficient.

Real-world problems commonly require the simultaneous consideration of multiple performance measures. Buying a new car is a simple illustration of such real-world multi-objective tasks. Comfort, price, depreciation factor, safety features, road tax and running costs such as fuel, service and repair are all criteria that usually car buyers consider and look to optimize when buying a new car. Most often, the multiple objectives are in conflict and compete with each other. Ultimately, the decision maker (DM) has to decide on an individual solution based on certain preferences and objective priorities. As an example, a DM might decide that the safety features in a car are prioritized over the running costs of its fuel consumption while on the other side the car's depreciation factor can be traded for its luxurious comfort.

1.1.1 Basic concepts

Unlike single objective optimization which aims to maximize, minimize or achieve a certain goal value for a single objective, multi-objective optimization consists of multiple criteria that need to optimize simultaneously. These criteria can manifest pair-wise relationships such as independence, harmony or conflict. In the former two relationships, an improvement or, alternatively deterioration in terms of a certain objective, will either have no influence on the performance of the remaining independent objectives or alternatively an impact of a similar nature. Such multi-objective optimization (MO) scenarios can be ultimately divided into a set of different single objective optimization problems in the case of complete independence, or reduced to a single optimization problem of one representative objective in the case of complete harmony. Optimizing multiple competing objectives is by far the most complicated multi-objective scenario, in such scenarios no single utopian solution can be found.

Definition: Multi-objective Optimization Problem

The Multi-objective Optimization problem in its general form can be described as follows:

$$\begin{array}{lll} \text{Minimize/Maximize} & f_m(x), & m=1,2,\ldots\ldots,M; \\ \text{Subjected to} & g_j(x) \geq 0 & j=1,2,\ldots\ldots,J; \\ & h_k(x) = 0 & k=1,2,\ldots\ldots,K; \\ & x_i^{(L)} \leq x_i \geq x_i^{(u)} & , i=1,2,\ldots\ldots,n. \end{array}$$

A solution x is a vector of n decision variables: $x = (x_1, x_2, \ldots, x_n)^T$. The last set of constraints is called variable bound, restricting each decision variable x_i to take a value within lower $x_i^{(L)}$ and an upper $x_i^{(U)}$ bounds. These bounds constitute a decision variable space Ɗ. Associated with the problem are J inequality and K equality constraints. The terms $g_j(x)$ and $h_k(x)$ are called constraint functions. A solution $x \in X$ that satisfies all the (J+K) constraint and the entire variable bound stated above is called a *feasible solution*. The set of all feasible solutions is called feasible region or search space S. For each solution **x** in the decision variable space, there exists a point in the objective space Z, denoted by $\mathbf{f(x)}$ = $z = (z_1, z_2,\ldots..,z_M)^T$. The mapping takes place between an n-dimensional solution space and an M-dimensional objective space. A maximization problem can be converted to a minimization problem by multiplying the function by -1[1].

Numerical analysis methods such as classical gradient descent, Newton-Fourier and Levenberg-Marquardt methods which operate in a single search space, termed as the decision variable space are use to solve single objective optimization. For thorough literature about numerical analysis and optimization techniques in the operations research (OR) community, reader is directed to Hillier and Lieberman in [2]. OR is an interdisciplinary science that deals with decision making, optimization, planning and coordinating activities of complex nature from the real-world.

In single objective optimization, a solution explored in the decision variable space replaces the current best solution only if it presents a superior objective function value. Operating in a single search space is yet another major difference with the simultaneous optimization of multiple objectives. Multi-objective optimization consists of finding the set of vectors in the decision variable space which produce the best set of solutions in the objective space. Usually, the search in the decision variable space is steered and influenced by the information that becomes available in the objective space.

In Fig 1.1 an optimization problem consisting of two objectives and three decision variables are illustrated. Note that the dimensionality of the objective space and the decision variable space can be any positive integer. The pink bounded area in the decision variable space denotes the feasible region of the space which is defined by certain application specific constraints. The objective vector function (f) maps a certain solution **‘x’** in the decision variable space to its corresponding objective vector. It is only through the objective function mapping that the performance of a certain candidate solution can be assessed.

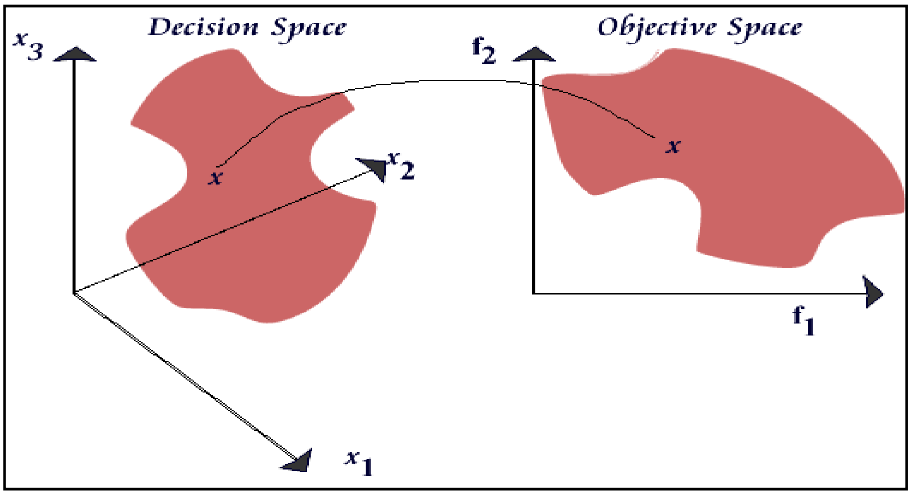

Fig 1.1 The multi-objective problem domain

Pareto Dominance

The idea of ‘optimal’ in MO can be traced back to the period between 1870 and 1900 with the work and philosophies of Ysidro Francis Edgeworth (1845-1926) [3] and Vilfredo Pareto (1848-1923), some of the most brilliant economists of the 19th century. Edgeworth main interests revolved around the utilitarian philosophy whose ultimate aim consisted of maximising society’s happiness by optimizing the problem of resource allocation. Vilfredo Pareto, on the other hand, concentrated on the use of classical programming techniques such as differential calculus and Lagrangian multipliers for the analysis of general equilibrium theories and the optimization of market efficiency. His work and theories constituted the foundation of the Pareto optimality concept which comprises the core of most multi-objective optimizers.

Definition: Pareto Dominance

A solution x in the search space S dominates a solution y in the search space S if both the conditions are true:

1. The solution x is no worse than y in all objectives, or

$f_j(x) \ntriangleright f_j(y)$ for all j=1, 2, …, M

2. The solution x is strictly better than y in at least one objective or

$f_j(x) \triangleleft f_j(y)$ for at least one j$\in$ {1,2,…, M}.

The superiority (or dominance) of one solution over the other cannot be established with many objectives in mind. When the following inequalities hold between two solutions x and y, it is said that solution y dominates the solution x:

$$f_i(x) \le f_i(y) \text{ for } \forall i \text{ and } \exists j\text{: } f_j(x) < f_j(y)$$

If the solution is not dominated by any other solutions, that solution is said to be a non-dominated solution. There exist many such non-dominated solutions in the search space known as *Pareto-optimal Solutions*. The curve formed by joining these solutions is known as a *Pareto-optimal Front* . The Pareto dominance concept is illustrated in Figure 1.2 in the objective space of a simple bi-objective scenario.

TABLE 1.1 OBJECTIVE VALUES IN2-D OBJECTIVE SPACE

Solution	f1 (maximize)	f2 (minimize)
1	14	5
2	13	3
3	12	2
4	10	1
5	10	4
6	6	3

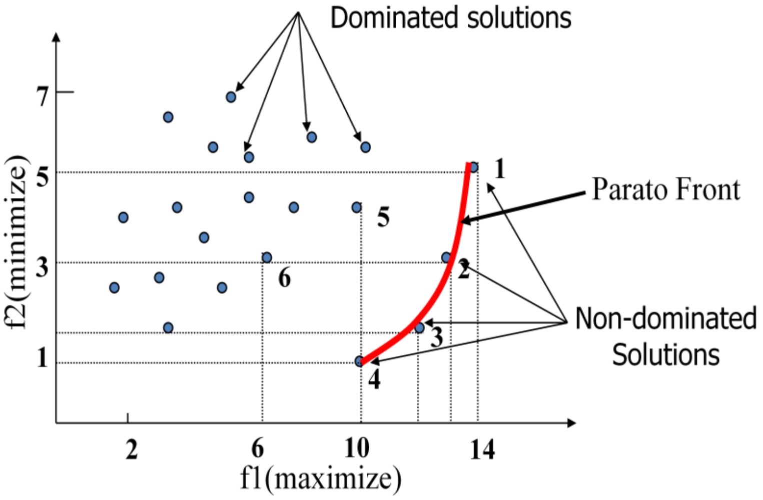

Fig 1.2 Pareto Dominance Illustration in a 2-dimensional objective space

Table 1.1 shows objective values of solutions in 2-D objective space. First objective function f1 is to be maximized and second objective function f2 is to be minimized. Fig 1.2 shows the plot of solutions given in table 1.1. Solution 1(f1=14, f2=5) is superior to solution 2 (f1=13) with respect to objective f1 and inferior than solution 2 (f2=3) with respect to objective f2. Solution 1 and 2 are non dominated solutions. In this way when all the solutions (1 to 6) are compared with each other it is found that 1, 2, 3 & 4 are non dominated solutions and 5&6 are dominated solutions. The line joining solutions 1, 2, 3 and 4 is called Pareto Front.

In the context of Pareto dominance, we can further distinguish between weak dominance and strong dominance or loose dominance and strict dominance respectively [1].

Weak dominance: Pareto dominance is sometimes simply referred to as a weak dominance. A solution x weakly dominates a solution y if x is better than y in at least one objective and is as good as y in all other objectives.

Strong dominance: A solution x strongly dominates a solution y if x is strictly better than y in all objectives.

Non-dominance: If neither x dominates y nor y dominates x, then both solutions are said to be incomparable or non-dominated. In this case, no solution is clearly preferred over the other, at least under the Pareto dominance criterion.

A set P* consisting of all non-dominated solutions for a given set of solutions P is called the non-dominated set of P. Any solution in P has not dominated solutions in P*. The non-dominated set P* is also termed as the Pareto set of P which refers to the decision variable space. In the literature, there is also another term known as the Pareto front of P which refers to the objective functions space.

Definition: Pareto Set (PS)

For a given MOP and its set of solutions P, the Pareto set P* is defined as follows:

$$P^* = \{x \in P \mid \neg \exists y \in P : y \succ x\}$$

When the set P is the entire search space (P = S), the Pareto set P* of the set P is called the Pareto-optimal set.

Definition: Pareto optimal Set (PF)

For a given MOP and its search space S, the Pareto-optimal set PF is defined as follows:

$$PF = \{x \in S \mid \neg \exists y \in S : y \succ x\}$$

1.1.2 Classical methods

Operations Research (OR) is a branch of mathematics within which a variety of techniques have been developed to deal with MOPs. These approaches developed within OR for solving MOPs are known as *classical methods*. Up to 1980 most of the computational methods to solve MOPs consisted of minimizing only one function, either using the other objective functions as constraints of the problem, or simply by taking a combination of all the objectives. The most common way to tackle a MOP is by *scalarization* which means reducing the problem to a single objective optimization problem (SOP). One example of this approach is the following method:

Weighted sum method: This method consists of transforming the vector of function values into a scalar value using an aggregating function over the vector function, getting the following problem:

$$\begin{aligned} &Minimize \quad g_w = \sum_{i=0}^{k} w_i f(x) \\ &subject\ to \quad 0 \langle w_i \quad for\ all\ i \in \{1,....,k\}; \\ &\qquad and \ \sum_{i=0}^{k} w_i = 1; \ \ x \in S \end{aligned}$$

In this way, the solution set consists only of one point for each weight combination. An important drawback of this approach is that controlling the weights dose not necessary help us to control the distribution of the points in the variable space. Besides, there are points that cannot be generated as a combination of weights in non convex cases—see [4] for a more in-depth explanation about this. There exist, in general, many scalarization methods which transform the MOP into a ‘classical’ SOP. It is worth to notice that by choosing a clever sequence of SOPs, a suitable finite size approximation of the entire Pareto set can be obtained (see [5] and references therein). Another approach to approximate the entire Pareto set is to use set oriented methods such as subdivision techniques [6].

ε-constraint method: In the ε *-constraint method* [7], one of the objective is chosen for minimization while the rest of the objectives conform a set of constraints limited by user specified bounds ε_i *i.e.:*

$$\begin{aligned} &Minimize \quad f_j \\ &subject\ to \quad f_i \le \varepsilon_i \quad for\ all\ i \in \{1,...,k\},\ i \ne j. \end{aligned}$$

The ε-constraint problem should be solved using multiple different values for ε_i if several Pareto optimal solutions are desired. This method can deal with convex and non convex functions; but, choosing the ε_i values is still an issue since there is no warranty that a feasible optimum exists for a specific ε_i. An in depth analysis of these method can be found in [8].

1.1.3 Requirements of optimizer

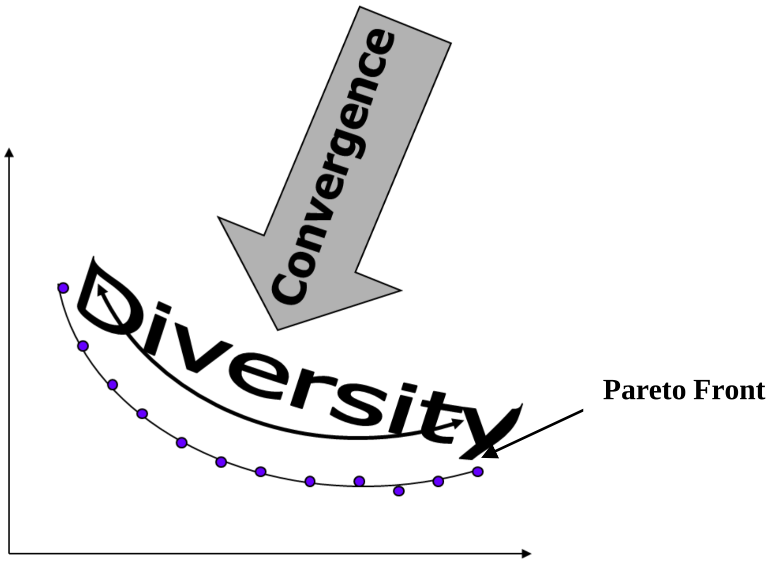

Fig 1.3 Two requirements of Multi-objective optimizer

The two requirements for multi-objective optimizers are usually sought and desired. They are shown in fig 1.3 and are as given below

Convergence: The approximation set achieved for a multi-objective optimization problem is required to as close as possible to the true Pareto front.

Diversity: Because of the non-existence of an ideal single solution in Multi-objective optimization frameworks with many competing objectives, and due to the fact that the global trade-off surface can potentially present an infinite number of solutions, the set of Pareto optimal solutions is also required to be well spread and uniformly covering wide areas of the Pareto front. Solutions diversity is conventionally preferred in the objective space as to present the DM with a well distributed set of solutions to choose from based on certain preferences such as objective priorities or region of interest (ROI). Solution's diversity is however not restricted to the objective space, and can be a desired requirement in the decision space of some applications.

1.1.4 Multi-objective Optimization Problems (MOPs)

Applying mathematics to a problem of the real-world mostly means, at first, modeling the problem mathematically, may be with hard restrictions, idealizations, or simplifications, then solving the mathematical problem, and finally drawing some conclusions about the real problem based on the solutions of the mathematical problem [9]. Real-world optimization problems involve a number of characteristics, which make them difficult to solve up to a required level of satisfaction. Those characteristics are

Existence of mixed type of variables (such as Boolean, discrete, integer and real).

Existence of non-linear constraints.

Existence of multiple conflicting objectives.

Existence of multiple optimum (local and global) solutions.

Existence of strong interaction among variables (epistatic).

Existence of stochasticities and uncertainties in describing the optimization problem.

Many real-world problems that could be transformed into optimization problems have complex search landscapes. These landscapes are unimodal or multi-modal, epistatic or non-epistatic and having strong local minima. Sometimes it is difficult to predict exact landscape of problem. Real-world optimization problem is a mix of difficulties mentioned above. Optimization problems are classified as

1. Single-objective problems (Have only one objective)
2. Multi-objective problems (Have 2 to 5 objectives)
3. Many-objective problems (Have more than 5 objectives)

Test suit: Due largely to the nature of multi-objective evolutionary algorithms (MOEAs), their behaviors and performances are mainly studied experimentally. In the past 20 years, several continuous multi-objective optimization problem test suites have been proposed the evolutionary computation community, which has played crucial role in developing and studying MOEAs. However, more test instances are needed to resemble complicated real-life problems and thus stimulate the MOEA research [10].

K. Deb in [11] has identified several features that may cause difficulties for multi-objective EAs in 1) converging to the Pareto-optimal front and 2) maintaining diversity within the population. Concerning the first issue, multimodality, deception, and isolated optima are well-known problem areas in single-objective evolutionary optimization. The second issue is important in order to achieve a well distributed nondominated front. However, certain characteristics of the Pareto-optimal front may prevent an EA from finding diverse Pareto optimal solutions: convexity or nonconvexity, discreteness, and nonuniformity. [12].

This section describes 15 multi-objective unconstrained (bound constrained) optimization problems used in the study. Test problems are chosen from a number of significant past studies in this area. First set of test problems contains Kursawe’s study (KUR) [13], ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 from ZDT test suit [12]. All problems have two objective functions and all objective functions are to be minimized. None of these problems have any constraint. Table 1.2 describes these problems. The table shows the number of objectives, number of variables, their bounds and the nature of the Pareto-optimal front for each problem.

Second set of test problems contains nine unconstrained (bound constrained) MOP test instances form the IEEE CEC09 algorithm contest for multi-objective evolutionary algorithms [10]. These test functions have characteristics, such as multi-modality and discontinuity, which generally cause difficulties to most MOEAs. Test problems under consideration are minimization problems.

TABLE 1.2 TEST PROBLEMS USED IN THIS STUDY

Problem	No. of Objective	No. of variables	Variable bounds	Comments
KUR	2	3	[-5,5]	Nonconvex
ZDT1	2	30	[0,1]	Convex
ZDT2	2	30	[0,1]	Nonconvex
ZDT3	2	30	[0,1]	Convex, Disconnected
ZDT4	2	10	$[0,1]x[-5,5]^{n-2}$	Convex
ZDT6	2	10	[0,1]	Nonconvex, Non-uniformly spaced
IEEE CEC09 Unconstrained (Bound Constrained) Test Problems				
UF1	2	30	$[0,1]x[-1,1]^{n-1}$	Convex PF
UF2	2	30	$[0,1]x[-1,1]^{n-1}$	Convex PF
UF3	2	30	$[0,1]^{n}$	Convex PF
UF4	2	30	$[0,1]x[-2,2]^{n-1}$	Non Convex PF
UF5	2	30	$[0,1]x[-1,1]^{n-1}$	Non Convex, Disconnected and uniformly spaced PF
UF6	2	30	$[0,1]x[-1,1]^{n-1}$	NonConvex, Disconnected and Non-uniformly spaced PF
UF7	2	30	$[0,1]x[-1,1]^{n-1}$	NonConvex PF
UF8	3	30	$[0,1]^{2}x[-2,2]^{n-2}$	Connected PF
UF9	3	30	$[0,1]^{2}x[-2,2]^{n-2}$	Disconnected PF

1.2 Evolutionary Algorithms

Evolutionary algorithms (EAs) are based on models of organic evolution, i.e. nature is the source of inspiration. They model the collective learning process within the population of individuals, each of which represent not only a search point in the space of potential solutions to a given problem but also may be a temporal container of a current knowledge about the “laws” of the environment [14]. The starting population is initialized by an algorithm-dependent method, and evolves towards successively better regions of the search space by means of (more or less) randomized process of recombination, mutation, and selection. The common opinion about EAs is that they explore the search space by the (genetic) search operators, while exploitation is done by selection. The environment delivers quality information (fitness value) for new search points, and the selection process favors individuals of higher quality to reproduce more often than worse individuals. The recombination mechanism allows mixing of parental information while passing it to their descendants, and mutation introduces innovation into the problem. EAs are population based, stochastic, and flexible, thereby providing an ideal platform to modify them to suit to solve most optimization problems. Thus, the task in an optimization process is to start from one or a few random solutions in the search space and utilize the objective functions and constraints to drive search towards the feasible region and finally reach near the optimum solution by exploring as small as a set of solutions as possible. Figure 1.4 provides an overview of computational intelligence and its components. Different sub-fields of evolutionary computing are also shown.

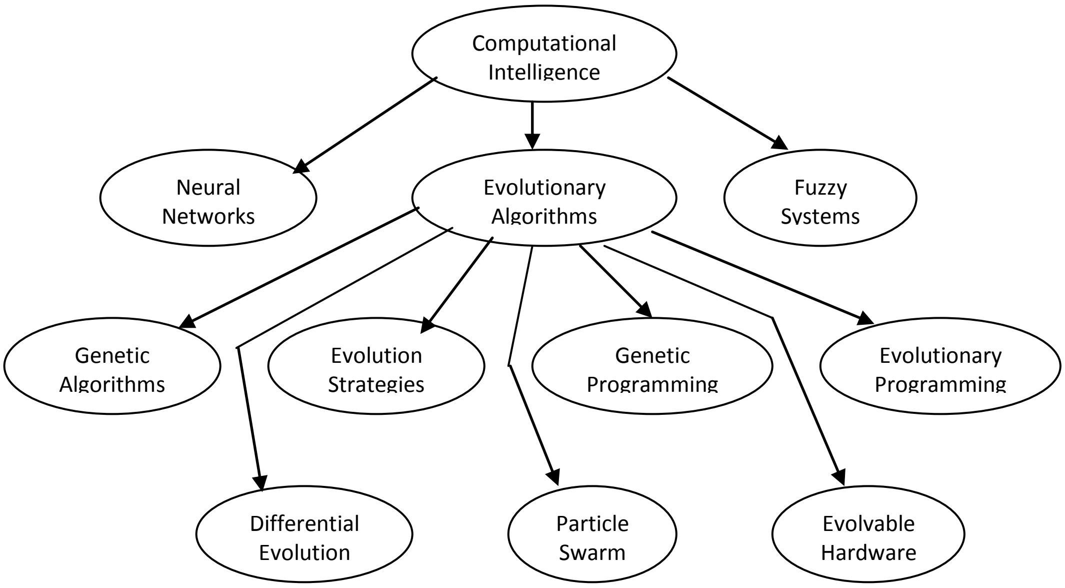

Fig 1.4: Computational intelligence and evolutionary computation

Number of researchers located in geographically distant places across the globe has suggested the idea of using evolutionary principles to constitute a computerized optimization algorithm. The origins of evolutionary algorithms can be traced to at least the 1950's. As of today, there are four dominant methodologies: Genetic Algorithms, Genetic Programming, Evolutionary Programming and Evolutionary Strategies. Besides the different historical roots and philosophy there are also technical differences between these streams in EA. These differences concern the representation applied, the corresponding genetic search operators, the selection mechanism and the role of self-adaptation. Evolution Strategies (“ESs”), like EPs, emphasize phenotypic transformations. GAs emphasizes the genotypic transformation of individual problem solutions. A typical GA represents a solution to a problem in terms of its genotypic features i.e. the basic features, or elements, that make up a solution. In the following section GA as a representative of EAs is discussed.

1.3 Genetic Algorithm

Evolutionary biologists who were explicitly seeking to model aspects of natural evolution has generated computer programs appeared in the late 1950s and early 1960s that might today be called genetic algorithms. By 1962, researchers such as G.E.P. Box, G.J. Friedman, W.W. Bledsoe and H.J. Bremermann had all independently developed evolution-inspired algorithms for function optimization and machine learning, but their work attracted little follow-up. As early as 1962, John Holland's work on adaptive systems laid the foundation for later developments; most notably, Holland was also the first to explicitly propose crossover and other recombination operators. J. H. Holland has strongly influenced the idea which appears first in 1967 in J. D. Bagley’s thesis “The Behavior of Adaptive Systems Which Employ Genetic and Correlative Algorithms” .Who is considered as the pioneer of genetic algorithms. However, the seminal work in the field of genetic algorithms came in 1975, with the publication of the book *Adaptation in Natural and Artificial Systems*. The book has described earlier research and papers by Holland and his colleagues at the University of Michigan. This book presents systematic and rigorous concept of adaptive digital systems using mutation, selection and crossover by simulating the processes of biological evolution as a problem-solving strategy [15]. Genetic algorithms are stochastic, population-based search and optimization algorithms loosely modeled after the paradigms of evolution. Genetic algorithms guide the search through the solution space by using natural selection and genetic operators, such as crossover and mutation. In the early to mid-1980s, genetic algorithms has been applied to a broad range of subjects, from abstract mathematical problems like bin-packing and graph coloring to tangible engineering issues such as pipeline flow control, pattern recognition and classification, and structural optimization. The spectrum of GAs applicability has gone well beyond the use of it as search and optimization algorithm. Now GAs have been used for automatic programming, for understanding biological systems, for modeling communities in ecologies, for modeling social organizations, and also as computational models of innovation and creativity.

1.3.1 Working of GA

Genetic algorithm (GA) is an iterative optimization procedure. GA works with a number of solutions (collectively known as a population). A flowchart of the working principle of a simple GA is shown in Fig 1.5. In the absence of any knowledge of the problem domain, a GA begins its search from a random population of solutions. If a termination criterion is not satisfied, reproduction and variation operators (crossover and mutation, and others) are applied to update the population of solutions. An iteration of these operators is known as a generation in the parlance of GAs. The name Genetic Algorithm derived from the fact that: the representation of a solution in GA is similar to a natural chromosome and GA operators are similar to genetic processes.

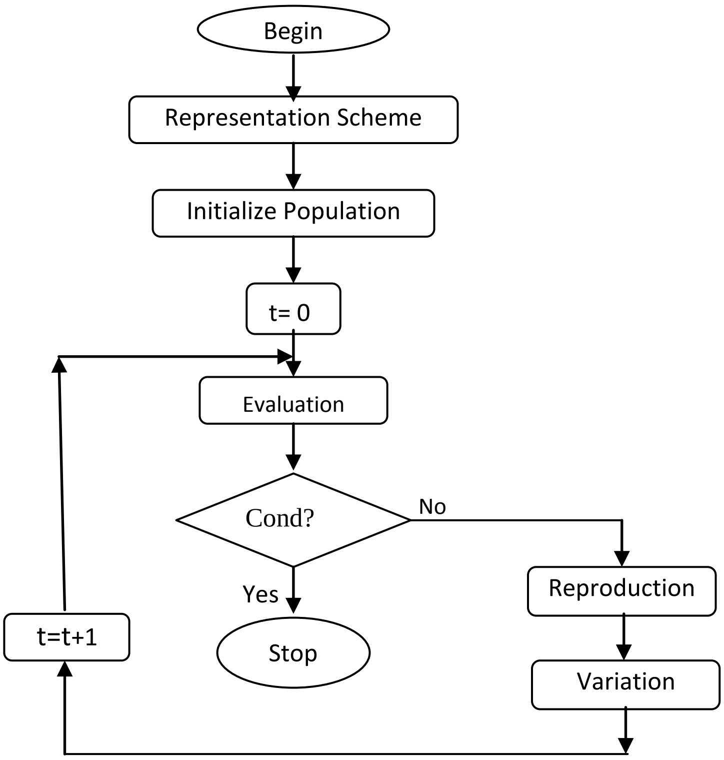

Fig 1.5 A Flowchart of working principle of a genetic algorithm.

The solution vector can be represented as a vector of real numbers, discrete numbers, a permutation of entities or others or a combination, as suitable to the underlying problem. If a problem demands mixed real and discrete variables, a GA allows such a representation of a solution. The purpose of variation operators is to create new and hopefully improved solutions by using the mating pool formed by the selection operator. Crossover is a very powerful tool for introducing new genetic material and maintaining genetic diversity. The selection process drives the search towards the regions of the best individuals. The mutation operator is an exploratory operator that reestablishes the genetic diversity loss during the evolution and explores new solutions avoiding premature convergence to local optima. Every search algorithm must establish a balance between two conflicting features: exploitation of the best solutions, depth search, and exploration of the search space, width search. GAs are general-purpose search methods where the selection process and the crossover and mutation operators establish a balance between the exploration and exploitation of the search space very adequate for a wide variety of problems.

The data type or representation is one of the distinguishing features of the different styles of GAs. In traditional GAs binary representation is used *i.e.* individuals or chromosomes are bit-strings with a fixed length. The study of sequencing problems such as routing and scheduling has yielded order-based GAs where each individual is a permutation. Parameter optimization problems with variables over continuous domains have led to real-valued or floating point GAs. Genetic Programming uses individuals that are tree structures.

It is argued earlier that depending on the fitness landscape and the position of the parent population, the population variance may decrease or increase after the reproduction operation. Thus to avoid any premature convergence or stagnation, the population after variation operation must adjust its variance accordingly, so as to keep the overall population variance to be of reasonable value from generation to generation. Particularly, if the reproduction operator reduces the variance of population, the variation operator must increase the variance of population and vice versa.

1.3.1.1 Individual Representation

Within the EA framework, individuals are considered at two levels: the genotypic level and the phenotypic level [16]. The genotype of an individual, also referred to as the chromosome, is a string of genes of finite length. The chromosomes could be in any form such as binary, integer/alphabet, real-valued. The most common forms are binary (binary representation) and integer/alphabet (e. g. permutation representation). In the binary representation, each gene in encoded individuals takes a value of either 0 or 1 and the value of genes is important. The coding of the variables is called genotypes, and the variables themselves are called phenotypes In the permutation representation, each gene in encoded individuals takes a distinguished value and the position of genes is important. In real-valued representation, chromosomes are represented as real-parameter vector. In this study GA with real-valued representation (also called Real-Coded GA (RCGA)) is used. The choice of chromosome representation, which depends on the problem as well as the EA itself, is important. An unsuitable representation could lead to unnecessary computational overhead and low performance of the EA.

1.3.1.2 Evaluation of Individual Fitness

The evaluation of an individual is the process which obtains the objective value, the phenotype of the individual by decoding its genotype. The phenotype of individuals could be used to directly compare individuals based on Pareto dominance and at different stages of the algorithm. However, it is very common that the algorithm takes one step further in assessing individual’s quality by assigning fitness to each individual. This process is referred to as the individual's fitness evaluation to distinguish from the individual's objective values evaluation. The fitness evaluation derives the fitness of an individual from its objective values. While computing the fitness of an individual, the fitness evaluation might or might not consider other individuals in the population. The fitness evaluation strategy is one of the core features of which could differentiate one EA from others. Fitness assignment strategies could be categorized into three types:

Dominance-based: The fitness of individuals is determined by comparing individuals to others in the population based on Pareto dominance (or other types of relaxed Pareto dominance); for example, NSGA-II [17] and SPEA2 [61].

Non dominance-based: The fitness of individuals is determined by applying transferring functions on the objective values of individuals which combine and/or modify the objective values; for example, MOEA/D.[19]

Hybridization: Based on some satisfaction on dominance condition, transferring functions are applied to the objective values of individuals to obtain the fitness value. For example, IBEA[62].

1.3.1.3 Parent Selection

The parent selection, or mating selection, is the process of selecting individuals to participate in the production of offspring [16]. There are two common mating selections: stochastic selection and tournament selection. Stochastic selection selects parents at random regardless of parents' quality/fitness. Tournament selection applies an additional layer onto the stochastic selection. In tournament selection, parents are also drawn at random but selected parents then undergo a fitness comparison process. The highest quality parent, i. e. the winner of the tournament, is selected to be in the mating pool. The size of the tournament is usually set to 2 (binary tournament). There are other schemes such as fitness proportionate selection and truncation selection. In fitness proportionate selection, the probability of each individual being selected for the mating pool is in proportion to its fitness.In truncation selection, each of the top (1/c)% individuals in the population (with respect to fitness) get c copies in the mating pool. The selection of parents is also one of the key features in EAs.

1.3.1.4 Replacement Strategy

The replacement strategy, which is also known as the environmental selection or survival selection, is the process of selecting individuals for the next generation based on the fitness of individuals. As opposed to the parent’s selection, which is usually stochastic, the replacement strategy usually select the best individuals based on their fitness, is mainly deterministic [20]. The replacement strategy is categorized into generational and steady-state selection schemes. The difference between these two schemes is at which point parents and offspring compete for survival. In the generational selection scheme, for each generation, the offspring population is constructed (based on the parent population) first. The sizes of the offspring population and parent population are usually the same size of the current population. The offspring population and the current population are combined. Individuals from resulting population are then selected for the next generation. However, the steady-state selection scheme allows offspring to compete for survivor and reproduction as soon as they are created. In other words, offspring are considered to replace their parents (or other individuals) in the current population immediately after offspring are constructed. Besides the fitness assignment and the parent's selection, the replacement strategy is also one of the important aspects of an EA. This thesis will discusses various replacement or parent selection schemes for formation of next generation.

1.3.1.5 Recombination Mechanism

The recombination mechanism uses the genetic material, the genotype, of parents to create offspring. The aim of the recombination mechanism is to manipulate the gene structure of individuals in the current population, create new individuals with the expectation that better individuals could be obtained. There are a large number of genetic operators that can be used for the recombination, but broadly these are divided into two categories:

Mutation: is a unary operator which is applied to the genotype of one individual in order to slightly modify the gene structure of that solution. The mutation operator is most times a stochastic operator. The mutation operator attempts to introduce a few new features that might not be inherited from the parents. The purpose is to add diversity to the population and contribute so that the entire search space can be possibly explored. The simplest form of mutation operator is bit lip mutation used with a binary representation. Each gene in the genetic material of an individual is inverted (between 0 and 1) with certain probability (usually very low). This probability is known as the mutation probability, which is relatively small to prevent too much disruption of the inherited genetic material [1].

Crossover Operator: is usually a binary operator which combines the genetic material from two parents to produce the genetic material for one or two offspring. The crossover operator, which is usually a random operator, decides which part of the genetic material from parents is inherited by the offspring. The principle behind the recombination is that "good" genes in parents are combined in the genetic material of the offspring.

It is noted that both crossover and mutation operators are genotypic representation dependent. Different individual representations often require different operators. It is also noted that while operators for mutation are usually problem independent, many crossover operators are problem specific. Recombination, or crossover, operators have many different forms. Two common fortes of crossover for binary representation are k-point and uniform crossover. The simplest versions of k-point crossover are one-point crossover and two-point crossover. In one-point crossover, a crossover point is selected at random then genes on one side of the crossover point are exchanged between individuals. In two-point crossover, two crossover points are selected at random and genes between two crossover points are exchanged between individuals [1].

Recombination operator in real-parameter (real-coded) GAs directly manipulates two or more real numbers to generate one or more real numbers as offspring [1]. The detailed study of many recombination operators can be found elsewhere [1] and [21].

Eshelman and Schaffer introduced the notion of interval schemata for real-coded genetic algorithms and suggested a blend crossover (BLX-α) operator [22]. Deb and Agrawal developed the simulated binary crossover (SBX) operator that creates children solutions proportional to the difference in parent solutions [23]. In the design of SBX operator, the search power of single point crossover, used in binary coded GA is calculated and derived a functional relationship between probability distribution and spread factor. Polynomial probability distribution for contracting and expanding crossover is used. SBX-l [24] developed by Raghuwanshi et al uses lognormal distribution for contracting and expanding crossover.

The multi-parent recombination operator combines the features of more than two parents to generate offspring. Eiben introduced, scanning and diagonal crossover which is the generalized multi-parent recombination operators used in GA [25]. Ono and Kobayashi [26] introduced the unimodal normal distributed crossover (UNDX) that generates offspring using a normal distribution defined by the three parents. This operator has a feature of independence from the coordinate systems and is excellent in characteristics preservation. The UNDX has shown excellent performance in optimization of multi-modal and highly epistatic functions. Tsutsui and Ghosh [27] proposed the three types of multi-parent recombination operators: the center of mass crossover operator (CMX), the multi-parent feature-wise crossover operator (MFX), and the seed crossover operator (SX). They used the blend crossover (BLX-α) [22] as the base operator. Tsutsui *et al*. [28] proposed the multi-parent simplex crossover (SPX) that generates offspring by uniformly sampling values from the simplex formed by m (2≤m≤number of parameters +1) parents. Like UNDX, the SPX is also independent from the coordinate systems. Kita *et al*.[29] proposed the multi-parent extension of the UNDX to enhance its exploration ability. This operator uses multiple parents to create offspring solutions around the center of mass of these parents. Deb *et al.* in [30] proposed the vector based multi-parent crossover operator PCX (parent-centric recombination operator) and with Generalized Generation Gap (G3) model for a real coded steady state GA. The PCX allows large probability of creating a solution near each parent. They have shown that G3-PCX has better convergence than gradient methods on some unimodal functions. All these proposed operators are tested for performance evaluation on unimodal and multi-modal functions with/without epitasis among the parameters.

The multi-parent polynomial distribution recombination operator (MPX) [31] is a multi-parent extension of simulated binary crossover operator (SBX) [23] and the multi-parent lognormal distribution recombination operator (MLX) [31] is a multi-parent extension of SBX with lognormal distribution (SBX-l) [24]. MPX and MLX operators were used for single objective optimization. Investigation on suitability of MPX & MLX for multi-objective optimization will be part of this study.

1.3.1.6 Stopping Criteria

Stopping criteria define states in which the evolutionary search terminates and the best individuals are presented. Stopping criteria usually vary accordingly to the type of applications and the purpose of the studies. For theoretical studies, in which the purpose is to investigate the performance of newly proposed EAs for example, the stopping criteria are usually the number of evaluations (or generations) or the amount of execution time. However, the latter is rarely used due to its low reliability and dependence on other factors such as hardware, operating systems and programming languages. For real-world application (especially in real-time application), in which the computational time is limited, it is sensible to set the amount of execution time as the stopping criterion. There are other criteria such as on-line performance metrics which keep track of the improvement of population or best solutions until no improvement after a certain amount of time or evaluations. The number of generations could be also the stopping criterion of real-world applications.

1.3.1.7 Other Issues in EA

The core features of an EA have been discussed above. Although, when designing an EA, it is also worthwhile to pay attention to other issues in order to design a good performing EA. These issues include, but are not limited to:

Population Initialization & Reinitialization: The simplest approach is to generate random individuals for the initial population. However, it is possible and often advisable to use heuristics to construct (better) individuals. The heuristics are usually knowledge-based and problem dependent. Apart from being generated at the start of the search, new individuals could also be introduced during the search by heuristics instead of using reproduction operators as discussed above. This is known as reinitialization in which a part or whole population is reinitialized if the search stagnates.

Generational vs. Steady-State: As aforementioned, the replacement strategy could be either a generational or a steady-state approach, also referred to as generational or steady-state selection. Fitness assignment strategy drives the decision on which approach is deployed within an EA. If the fitness assignment strategy requires all (or a large number of) individuals in the population to estimate the fitness of an individual, the generational approach should be employed. On the other hand, the steady-state selection could be used if the fitness of an individual is not affected by other individuals in the population. Inappropriate approaches could lead to expensive computational time and degraded performance of the search. It is noted that recent studies show a better performance of the steady-state selection over generational selection. For example, Durillo et al. [32] modified NSGA-II and SPEA2 from generational to steady-state selection and reported better performance than the original algorithms although the computational time increased significantly. Recent steady-state selection EAs, such as MOEA/D [19] also report better performance than generational state-of-the-art MOEAs such as NSGA-II and SPEA2. Therefore in this study emphasis will be given to this issue.

Exploration vs. Exploitation. This is also known as diversity vs. intensification of the population. The reproduction mechanism is mainly the driving force for exploration whereas exploitation is taken care by the replacement strategy. In EAs, it is very difficult to obtain good results in terms of both exploration and exploitation at the same time. There is usually a trade-off between these two aspects. Therefore, balancing well this trade-off could lead to high performance MOEAs.

Elitism: This term means that the best individuals are always included in the next population. It is unambiguous in the single-objective framework where there is only one best individual at anytime. However under the multi-objective framework, there could be always more than one best individual. The number of best individuals could increase dramatically with respect to the number of objectives of the problem. With a limited population size, it is non-trivial to identify which best individuals should be kept. An external archive or favouring best solutions with at least one best objective value could be the answer. Laumanns et al. in [33] also argued that the usefulness of elitism strongly depends on the mutation rate. However, in the author's opinion, it remains inconclusive how elitism should be tackled.

Duplication: Duplication occurs in the population if there is at least a pair of distinct (in the decision space) individuals having the same objective values. The problem with allowing duplication is that all individuals in the population could converge too quickly to a single point in the objective space which is not desirable. To deal with this problem, a hard approach it to strictly not allow any duplication while in a soft approach is to allow the replacement strategy to eliminate duplication (like in NSGA-II and SPEA2).

Mating Restriction: The idea of restricted mating comes from natural selection where mating only seems to happen between similar individuals. Deb and Goldberg in [34] argued that mating between dissimilar individuals will likely lead to unfit offspring called lethals. However, mating between too similar individuals, known as in-breeding, could affect adversely the progress of the search. Mating restriction could be performed on either the genotype or phenotype of individuals.

1.3.2 Parameters of GAs

Parameters used in GAs are classified as

1. Algorithm dependent
    - Population size (N)
    - Offspring population size
    - Crossover Probability
    - Mutation Probability
2. Operator dependent
    - Distribution Index for crossover operator
    - Distribution Index for mutation operator
    - Number of parents participating in recombination operation
    - Number of Offspring generated by recombination operator

3. Problem dependent
    - Number of variables (n)
    - Range for variable initialization
    - Bound check for variables

1.4 EA for solving MOP

Evolutionary algorithms are a popular tool for multi-objective optimization. The mechanisms that underpin their special utility are described in Section 1.4.1. A brief history of the MOEAs is provided in Section 1.4.2. As discussed previously in Section 1.1.3 a multi-objective optimizer is required to produce an approximation set that is close to globally Pareto optimal and that contains a rich distribution of solutions in regions of interest to the decision-maker. Distinct EA components have been developed to address each aspect of approximation set quality. In Section 1.4.3, methods for obtaining good proximity to the global Pareto front are reviewed. In Section 1.4.4, methods for obtaining a suitable distribution are discussed.

1.4.1 Why use EA for solving MOP

In addition to the general benefits of using an EA as a search tool, the EA concept is particularly suitable for multi-objective tasks. The population-based nature of the algorithm permits objectives to be treated distinctly through the notion of Pareto dominance and permits a family of trade-off solutions to be produced in a single execution of the algorithm. The fundamental benefit of this latter factor over multiple-start strategies is the potential for a cooperative search for ultimately different solutions, thus saving on the total number of solution evaluations required. In multi-start strategies that rely on particular parameter settings to provide direction toward a particular area of the Pareto front, there is generally no guarantee that a good distribution of parameter settings will ultimately lead to a good distribution of solutions on the trade-off surface. The key MOEA benefit of not requiring objectives to be aggregated in some way to form an overall cost function cannot be over emphasized. It is generally very difficult to aggregate objectives in a manner that precisely captures the DM preferences. Also, the required normalization of non-commensurable objectives can be far from straightforward. The MOEAs Pareto-based approach offers flexibility and information-richness with regard to solution performance discrimination, and assists the DM in learning about the problem as the search progresses.

1.4.2 History

The first evolutionary algorithms that were purposefully designed to obtain an approximation set were proposed in the mid-1980s ([35], [36] and [13]). In these schemes, a proportion of the population was selected according to each individual objective.

The main difficulty with this approach is that it often creates a phenomenon known as speciation, in which solutions arise in the population that are particularly strong in a single objective and particularly poor in others. Thus, important compromise solutions remain undiscovered, since the recombination of solutions from different extreme regions of the tradeoff surface cannot usually be assumed to generate an ‘intermediate’ compromise.

In the weighted-sum approach to MO, performance is captured in a single objective, calculated as a weighted-sum of individual performance in each of the original individual objectives. The well-known drawbacks of this approach are the difficulty in setting values for the weights, and the necessary condition for convexity of the trade-off surface that is required to obtain all Pareto optimal solutions. Thus, no combination of weights exists that can generate solutions in non-convex regions of the trade-off surface, as shown geometrically by Fleming and Pashkevich [37]. However, MOEAs based on weighted-sums schemes have also been proposed. Haleja and Lin [38] included the weight vector in the solution genotype and allowed multiple weight combinations to be propagated through the population during evolution. Jin, Okabe and Sendhoff [39] varied the weight vector over the evolution, and have also provided theoretical justification for the method ([39] & [40]).

Unlike these early attempts, the majority of modern MOEAs are based on the concept of Pareto dominance [41]. The use of Pareto dominance as a basis for solution comparison in Goldberg first suggested EAs in [15], together with the use of a niching technique to encourage solution distribution across the trade-off surface. In the early-1990s, three much-cited techniques emerged based on Goldberg’s ideas:

Fonseca and Fleming’s [42] multi-objective genetic algorithm (MOGA), Horn and Nafpliotis’s [43] niched Pareto genetic algorithm (NPGA) and Srinivas and Deb’s non-dominated sorting genetic algorithm (NSGA) [44], although early less well-known implementations by Ritzel and Cieniawski have also been reported ([43]and [45]). The techniques differ slightly in the way in which fitness is derived from Pareto comparisons of solutions. MOGA, NPGA, and NSGA all use fitness sharing for diversity promotion [46].

In the late-1990s, new methods were proposed to improve on the performance of the earlier Pareto-based algorithms. The innovations were usually evaluated on bi-objective test problems. Research efforts have focused particularly on the selection-for-survival aspect of an MOEA, with new methods for preserving and using identified (relatively) good solutions. Techniques for population density estimation and its use in diversity-promotion schemes have also been the focus of contemporary research. These ideas have been implemented in algorithms such as Zitzler and Thiele’s strength Pareto evolutionary algorithm (SPEA) [18] , Corne, Knowles and Oates’s Pareto envelope-based selection algorithm (PESA) [47], and Deb, Pratap, Agarwal and Meyarivan’s elitist non-dominated sorting genetic algorithm (NSGA-II) [17].

1.4.3 Methods for convergence

The proximity (or otherwise the convergence) of an approximation set to the Pareto front is the primary requirement of evolutionary multi-objective optimization. In order to achieve proximity, the search process should be steered in the right direction towards the Pareto optimal front of a certain multi-objective problem. This steering is more accurately achieved through the selection processes that govern MOEAs. As a result, fitter solutions, hence closer to the Pareto front, have higher chances for being selected for contributing to the next generations through the variation operators. Additionally, at the environmental selection process, fitter solutions would equally have higher chances for being selected for survival to the next generations.

Based on the Pareto dominance scheme, several techniques for ranking candidate solutions and assigning them fitness values were proposed for MOEAs. The interested reader is also referred to Zitzler, Laumanns and Bleuler [48] for more information about ranking and fitness assignment in MOEAs. The fitness values assigned to alternative candidate solutions to a MOP usually reflect their relative performance, and hence their prefer ability in terms of closeness to the Pareto front. These fitness values are then used as the primary selection criteria for variation and contribution to the next generation.

Elitism is a strategy which aims to ensure that any good solutions found during the optimization do not get filtered out and lost [48]. The implementation of an elitist strategy can be achieved by deploying an active selection for survival strategy (similar to the (μ+ λ)-ES), or using an external archive of non-dominated solutions. The latter elitist approach can be implemented in two different ways. An archive can be offline (or inactive) or otherwise online (or active). The first generation of MOEAs (e.g. MOGA) used to deploy offline archives to store all the non-dominated solutions achieved all along the optimization process. The offline archive however did not have any impact on, or interaction with, the evolutionary search. On the other hand, some of the second generation of MOEAs (e.g. SPEA2) deploy an online archive as an elitist strategy. In addition to storing the best ‘representative’ solutions, the content of the online archive is used to steer the search by participating in the mating procedures and contributing to the next generations. This last strategy of elitism is generally more efficient then its counterpart (deploying offline archives) and results in an accelerated convergence towards the Pareto front [1].

Moreover, it is worth mentioning that the majority of the first generation of MOEAs used to deploy a (μ, λ) strategy at the selection for survival stage. All the offspring solutions deterministically replaced their parents regardless of their performances. In other words, the selection for survival process can be described as inactive in the majority of the early approaches and hence elitism was absent. Deploying elitism in the form of an external archive, the inclusion of candidate solutions in the archive is performed in *en bloc* or *incremental* fashion. The latter two terminologies suggested by Zitzler in [18] and correspondingly denoted the strategy of simultaneously including a set of solutions into the external archive (e.g. SPEA2) or otherwise the inclusion of one solution at a time (e.g. PAES, PESA). While in the latter approach, the order of including the solutions in the archive is essential, in en bloc strategies the order of inclusions is irrelevant.

The importance of elitism was particularly highlighted after the outcome of several critical studies ([18] and [12]). These studies were conducted in the aim of contrasting the performance of multiple MOEAs on a set of bi-objective optimization problems. Some of the MOEAs included in the comparative studies deployed elitist strategies, while other MOEAs did not. The studies' outcomes illustrated that the elitist MOEAs were generally outperforming their counterparts with no active elitist strategies. Indeed, study has shown that in order to ensure the convergence of the population of solutions handled by a MOEA in the limit, elitism is an essential and a theoretical requirement [49].

1.4.4 Methods for Diversity

Fitness sharing, a concept first introduced in 1989 by Goldberg in [15], is one of the earliest attempts for promoting diversity as a requirement in EMO. Fitness sharing is based on a density estimation approach and was originally motivated by the need for *niche formation* to prevent premature convergence towards sub-optimal regions of the objective space. As its name reveals, the 'fitness sharing' method forces the sharing and therefore the degradation of the fitness values corresponding to solutions lying within a certain distance from each other. The notion of distance is an application dependent variable, and is usually the Euclidean distance in most real-coded applications. The process of sharing fitness values commonly penalizes solutions populating dense areas of the search space without violating the Pareto dominance notion. In other words, fitness sharing discriminates each set of solutions belonging to a certain level of performance or rank in terms of diversity without degrading their membership with the rank they occupy. Hence, a non-dominated solution lying in a poorly populated area of the objective or decision space will always have the highest probability of selection for variation or survival. The major disadvantage of the fitness sharing method is that its success is highly dependent on the chosen niche size parameter.

Several alternatives to the fitness sharing technique for density estimation were proposed in MOEAs. The most widespread alternatives include the *Nearest Neighbor* approach and the *histogram* based techniques. In a framework aiming to improve the performance of SPEA [18], Zitzler et al suggested a density estimate based on the use of the *kth* nearest neighbor measure in the Euclidean objective-space. A statistical heuristic was deployed to determine the value of the critical parameter *k* based on the square root of the population size. Other density estimation approaches based on the nearest neighbor included the method suggested by Abbass, Sarker and Newton (2001) [50] who replaced the use of the *kth* nearest neighbor measure as a density estimate with the mean Euclidean distances of the *two* nearest solutions. Sarker, Liang and Newton (2002) [51] later on extended the technique by Abbass *et al* to incorporate the mean Euclidean distances of the *M* nearest solutions as the density estimate. The two most popular diversity-preserving operators based on the nearest neighbor density estimation are the clustering technique [18] and the crowding technique [17].

Despite their conceptual simplicity and their relatively low computational requirements, the nearest neighbor diversity promotion mechanisms suffer from a fundamental disadvantage due to their requisite for the consistency and scalability of potentially no commensurable objectives. Another disadvantage of the nearest neighbor approaches is their requirement for a sensitive choice of the parameter *k* which is essential to the success of the technique. This disadvantage is vaguely similar to the choice of σ_{share} in the fitness sharing approach for promoting diversity.

On the other hand, histogram based techniques are an alternative density estimation procedure that overcome the disadvantage of requiring distance measurements and the concatenation of non-commensurable objectives in the nearest neighbor schemes. These alternative techniques operate by partitioning the objective space into a grid of different hyper boxes. The density estimation is then based on the count of the number of solutions residing in a certain hyper box in the objective space. Several objective space gridding systems where introduced for MOEAs. In the context of the Pareto archived evolution strategy (PAES) [47] by Knowles and Corne, the user specifies the number of bisections in terms of each objective range, and therefore specifies the spacing between the objective space hyper boxes. These user-specified spacing, ranges between partitions of the objective space which are then used in an adaptive grid spacing system defined by the locally non-dominated solutions. Deb *et al* suggested a steady state approach (ε-MOEA)[52] based on a combination of the ε-dominance concept introduced by Laumanns *et al* [53] and an adaptive grid archiving (ADA) similar to PAES. Laumanns *et al* [53] used the ε-dominance concept to implement a histogram-based diversity promotion strategy based on the proposition by Papadimitriou and Yannakakis (2000) [54]. ε-MOEA maintains diversity in the archive by allowing only one solution in each pre-assigned hyperbox in the objective space. No specific upper limit on the archive size is needed to be predetermined as the archive gets bounded according to the chosen ε vector in terms of each objective. Despite their increasing popularity, one of the major drawbacks of histogram-based diversity promotion techniques is their exponential computational complexity in the number of objectives. The gridding system applied in the objective space is yet another sensitive design parameter which can be inappropriate to some Pareto front structures. Choosing the right gridding system is particularly a hard design choice when the dimensionality of the objective space increases.

Other miscellaneous approaches for promoting diversity include *mating restrictions*, *lateral diversity* and *target vector* approaches. Deb and Goldberg [34] suggested mating restriction after a study on multimodality. They have noticed that when solutions from remote areas of the search space (objective or decision space) recombine, they most often produce week offspring known as *lethals*. Booker (1982) was the first to introduce the restrictions on the mating process for promoting diversity in the approximation set [34]. Similar to fitness sharing's parameter σ_{share} , a parameter σ_{mate} is calculated to determine whether two selected solutions belong to different regions of the space and should be restricted from mating. Lateral diversity on the other hand, suggests maintaining diversity in the dominated regions of the space for obtaining an overall better diversity in the population. This is interpreted as a requirement for keeping a balanced tradeoff between diversity and proximity to the Pareto front [55]. Examples of studies deploying lateral diversity for maintaining a tradeoff between proximity and diversity include Deb and Goel [56], Laumanns et al.[57], Laumanns and Ocenasek [58], and Bosman and Thierens [55].

Moreover, target vector approaches, originally suggested in the OR community and used to assess the performance of EAs, are deployed as a diversity promotion techniques in MOEAs. The approach consists of suggesting a target point in the search space, and then seeks to minimize the distance between candidate solutions and the suggested target point [59]. Later on in the MOEAs, the target vector approach was extended to allow the suggestion of multiple target objective vectors [60]. In the extended approach, better fitness scores get allocated to solutions close to certain target vectors. Nevertheless, fitness sharing takes place when multiple solutions reside in the neighborhood of a target objective vector in the aim of promoting diversity and avoid the genetic drift towards a single target vector.

1.4.5 Performance Measures

An important aspect of an algorithm is its performance, i.e., how “good” it is in carrying out a specific task. For such a characterization one has to evaluate the quality of the result in relation to the resources that were needed to achieve it. The notion of performance includes both the quality of the outcome as well as the computational resources needed to generate this outcome. Concerning the latter aspect, it is common practice to keep the number of fitness evaluations or the overall runtime constant in this sense, there is no difference between single and multi-objective optimization. As to the quality aspect, however, there is a difference. In single-objective optimization, we can define quality by means of the objective function: the smaller (or larger) the value, the better the solution. In contrast, there are two goals in a multi-objective optimization: To find a set of solutions as close as possible to the Pareto-optimal front and to find a set of solutions as diverse (or non-dominated) as possible

There are several metrics that have been proposed to measure the performance of MOEAs. Some of the most popular performance metrics used in this work are described below:

Inverted Generational Distance: Let Q be non-dominated set and P* be Pareto optimal set. This metric indicates how close the Pareto-optimal front is to nondominated solutions set is.

$$IGD(Q, P*) = \frac{\sum_{v \in P*} d(v, Q)}{|P*|}$$

Where d(v,Q) is the minimum Euclidean distance between v and the points in Q. IGD measures both convergence and diversity of Q. The lower the value of the IGD, the more diversity in the non-dominated solution set indicating the better performance of the algorithm. [10]

SPREAD (Δ) metric: The Spread indicator is a diversity metric that measures the extent of spread achieved among the obtained solutions. This metric takes a zero value for an ideal distribution, pointing out a perfect spread out of the solutions in the Pareto front. It is desired that the Pareto set be well spread. Euclidean distance between any two neighbor solutions in non-dominated solution set is calculated and then the average of these distances is obtained. The non-uniformity in the distribution, Δ, is calculated as:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} \left| d_i - \bar{d} \right|}{\sum_{m=1}^{M} d_m^e + |Q| \bar{d}}$$

Where d_i can be any measure between neighboring solutions and $\bar{d}$ is the mean value of these distance measures. The parameter d_m^e is the distance between the extreme solutions of P* and Q corresponding to m-th objective function. [1]

Two Set Coverage (SC): This metric is used for comparison of two sets of nondominated solutions. Let X be the set of decision vectors for the considered problem and A,B$\subseteq$X are two sets of decision vectors. The function SC maps the ordered pair (A,B) into interval [0,1] :

$$SC(A,B)=\frac{\left|\{b\in B/\exists a\in A:a\geq b\}\right|}{|B|}$$

If all solutions in A dominate or are equal to all points in B then by definition SC = 1. SC = 0 implies the opposite. In general SC(A,B) and SC(B,A) both have to be considered due to set intersections not being empty. This metric can be used for decision as well as objective space. In this work we have used it in objective space.[12]

Statistical test: A popular way to compare the overall performances of algorithms is to count the number of cases on which an algorithm is the overall winner. Some authors also use these counts in inferential statistics, with a form of two-tailed binomial test that is known as the sign test. In this study sign test is used as Statistical test to compare performance of proposed algorithm with other algorithm. The difference of performance scores of the two algorithms on a problem should be significant. The direction of any significant differences is denoted as follows:

A plus sign (+): the performance of the proposed algorithm is better than or comparable with the one of the corresponding algorithm

A minus sign (-): Performance of proposed algorithm is insignificant.

CPU Time: CPU Time measure shows computational efficiency of proposed algorithm.

It is noted that several performance metrics, which require the true Pareto front Point for calculation, include the generational distance, the inverted generational distance and the distance to the Pareto optimal front. However, it is often true that the Pareto front is unknown for a given problem. In this case, an estimated true Pareto front for that problem could always be used. The estimated true Pareto front could be obtained by using integer programming to solve the problem (with long computational time), solving a relaxed version of the problem, or in the worst case by combining the best solutions from several runs obtained by all algorithms under investigation.

It is usually the case that a number of performance metrics are used in conjunction to assess the performance of MOEA. The reason behind it is that within the multi-objective optimization framework there are several criteria to assess the MOEA performance such as diversity, convergence and distribution of an obtained set of solutions. One performance metric is often only able to assess on of such criteria. Furthermore the criteria are normally conflicted. Performance metrics we will use in the study are IGD, SPREAD, SC and CPU Time to asses' performance of algorithms. Also some statistical tests will be performed for performance comparison.

1.5 Summary

In this chapter we have introduced basic concepts of multi-objective optimization, classical methods of solving multi-objective problems and requirements of a multi-objective optimizer. Multi-objective optimization problems and their characteristics have been discussed along with few problems from commonly used test suits like ZDT and CEC09. Genetic algorithms are briefly introduced, including the terminology, working principles, major components, and algorithmic flow controls. After this GAs suitability for solving multi-objective optimization problem has been discussed. An overview of multi-objective genetic algorithm has been presented and various methods for convergence and diversity preservation are reviewed. Finally, issue of performance measure of multi-objective genetic algorithm is addressed. The performance metrics this study will be using to measure performance of proposed multi-objective genetic algorithms have been described in this chapter.

Chapter 2

NSGA-II with Multi-parent Recombination Operators

The concept of Multi-objective optimization and Genetic Algorithm is explained in previous chapter. This chapter describes the Non-dominated Sorting Genetic Algorithm II (NSGA-II). This algorithm is also known as the state-of- the-art MOEA. NSGA-II uses nondominated sort and crowding comparison operators along with SBX crossover operator. This chapter also describes two multi-parent crossover operators, MPX (multi-parent crossover with polynomial distribution) and MLX (multi-parent crossover with lognormal distribution). The encouraging results obtained by these operators on single objective problems as given in the study [31] by Raghuwanshi et al, has motivated us to test them on multi-objective optimization problems. This chapter also presents an attempt to improve performance of real coded NSGA-II by replacing SBX operator with MPX and MLX operators. This study also investigates suitability of multi-parent operators for solving MOPs.

2.1 Overview of NSGA-II

K. Deb *et al.* [17] has proposed the *Elitist Non-dominating Sorting Genetic Algorithm* (NSGA-II) and, due to its proven robustness and efficacy, it has been widely used as a reference to assess the performance of new MOEAs. It has remarkable differences with its predecessor, the Non-dominating Sorting Genetic Algorithm (NSGA) [44], other than the addition of an elitism mechanism.

NSGA-II successfully combines the following key elements:

1. A fast-nondominated sorting approach.
2. A density estimator.
3. A crowded comparison operator.
4. Recombination operator
5. Mutation operator

Pseudo-code of NSGA-II is given below:

```
1.  Procedure NSGA-II (gen)
2.  Randomly generate population of individuals $P_t$, ($t=0$)
3.  Sort the population into a set of different domination levels by using nondominated sort
4.  Assign the fitness to each individual as its non-domination level
5.  Create an offspring population $Q_t$, of size N, from $P_t$ by using tournament selection, crossover and mutation operators
6.  for t= 1 to gen do
7.      Combine parent and offspring population to conform
8.      $R_t = P_t \cup Q_t$
9.      Perform non-dominating sorting to $R_t$ and identify different fronts $F_i$, $i = 1, \ldots$, etc
10.     Set new population $P_{t+1} = \phi$, set a counter $i = 0$
11.     repeat
12.         $P_{t+1} = P_{t+1} \cup F_i$
13.         $i = i+1$
14.     until $|P_{t+1}| + |F_i| < N$
15.     Perform the crowding-sort ($F_i$) procedure and include most widely spread $N - |P_{t+1}|$ solutions by using crowded distance values in sorted $F_i$ to $P_{t+1}$
16.     Create an offspring population $Q_t$, of size N, from $P_t$ by using crowded tournament selection, crossover and mutation operators
17. end for
18. end procedure
```

In the above algorithm lines 3 and 4, refer to the process known as *nondominated sorting;* this process consists of classifying the population into several disjoint layers F_i ‘*s* (non-dominated sets)*;* such that $P = \cup F_i$. The main feature of these classes F_i is that any two members of the same class are incomparable in the Pareto sense. As a second criterion for ordering, after considering the Pareto rank of each solution, the *crowding* distance value is used (Line 15). Crowding distance is an indicator of the density of individuals around a particular individual p_i inside the population. For this, the average distance of two solutions, on either side of solution p_i*;* is taken along each objective. To create an offspring population, binary tournament selection followed by recombination and mutation is used. In this case, a modification called *crowded tournament* is employed for selection. In this operator, a solution p_i wins a tournament against other solution p_j if p_i has a better (smaller) Pareto rank or, in the case that both have the same rank value, when p_i has a better crowding distance value. This last condition provides a way to proceed in case of incomparable solutions, and, in terms of the evolutionary process, it helps to maintain diversity.

A brief description of non-dominated sorting procedure, density estimation procedure, crowded comparison operator and SBX crossover operator used in NSGA-II are given below.

A fast non-dominated sorting approach: In order to sort a population of size N according to the level of non-domination, each solution must be compared with every other solution in the population to find if it is dominated. First, all individuals in the first nondominated front are found. In order to find the individuals in the next front, the solutions in the first front are temporarily eliminated from the population. The procedure is repeated to find all the subsequent fronts. The fast non-dominated sorting procedure returns a list of the non-dominated fronts. Procedure for the same is given below:

```
fast-non-dominated-sort(p)
for each $p \in P$
    for each $q \in P$
        if $(p \prec q)$ then                    if p dominates q then
            $S_p = S_p \cup \{q\}$                include q in $S_p$
        else if $(q \prec p)$ then               if p is dominated by q then
            $n_p = n_p + 1$                       increment $n_p$
      if $n_p = 0$ then                          if no solution dominates p then
          $F_i = F_i \cup \{p\}$                  p is a member of the first front
i = 1
while $F_i \neq \phi$
   $H = \phi$
  for each $p \in F_i$                           for each member p in $F_i$
    for each $q \in S_p$                         modify each member from set $S_p$
     $n_q = n_q - 1$                             decriment $n_q$ by one
     if $n_q = 0$ then $H = H \cup \{q\}$         if $n_q$ is zero then q is a member of list H
i = i + 1
$F_i = H$                                        current front is formed with all members of H
```

Density Estimation: To get an estimate of the density of solutions surrounding a particular point in the population the average distance of the two points on either side of this point along each of the objectives is adopted. The obtained quantity serves as an estimate of the size of the largest cuboids enclosing the point of interest, without including any other point in the population (the so-called crowding distance). The following algorithm is used to calculate the crowding distance of each point in the set c:

```
$Crowding - distance - assignment\ (I)$
$l = |I|$                                      number of solutions in I
 for each $i$, set $I[i]_{distance} = 0$             initialize distance
 for each objective $m$
    $I = sort(I, m)$                              sort using each objective value
    $I[1]_{distance} = I[l]_{distance} = \infty$     so that boundary points are always selected
    for $i = 2$ to $(l-1)$                        for all other points
      $I[i]_{distance} = I[i]_{distance} + (I[i+1].m - I[i-1].m)$
```

Crowded Comparison Operator: The crowded comparison operator guides the selection process at the various stages of the algorithm towards a uniformly spread out Pareto-optimal front. Let us assume that every individual /in the population has two attributes.

1. Non-domination rank
2. Local crowding distance

$$\begin{aligned}&\text{We define a partial order } \geq_n \text{ as}\\ &i \geq_n j \quad \text{if } (i_{rank} < j_{rank}) \text{ or } ((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance}))\end{aligned}$$

That is, between two solutions with differing non-domination ranks we prefer the point with the lower rank. Otherwise, if both the points belong to the same front then the algorithm prefers the point which is located in a region with lesser number of points.

NSGA-II has made use of SBX crossover operator for recombination. In 1995, K.Deb and his students developed the Simulated Binary Crossover operator (SBX). First, they have calculated the search power of the single-point crossover operator of binary coded GA and later the SBX operator is developed to have the similar search power (The search power of a crossover operator is defined as a measure of how flexible the operator is to create an arbitrary point in the search space.). The procedure of computing the children solutions y1 and y2 from parent solutions x1 and x2 is described as follows.

A spread factor β is defined as the ratio of the absolute difference in children values to that of the parent values:

2.

$$\beta = \left| \frac{y2 - y1}{x2 - x1} \right| \tag{2.1}$$

3.

The spread factor β signifies a spread of children points relative to that of the parent points.

On multi-variable problem, the SBX operator performs variable-wise crossover using polynomial probability distribution.

The working of the SBX operator is described as follows. In multi-variable crossover, for each variable that undergo crossover (depends upon the crossover probability), generate a random number u_i between 0 and 1. From the polynomial probability distribution function, find the ordinate β_i so that the area under the probability curve from 0 to β_i is equal to the u_i. The probability distribution is given as

$$P(\beta) = \begin{cases} 0.5(\eta + 1)\beta^{\eta} & \text{if } \beta \leq 1 \\ 0.5(\eta + 1)\dfrac{1}{\beta^{\eta + 2}} & \text{otherwise} \end{cases} \tag{2.2}$$

In the above expression, the larger value of distribution index η offers a higher probability for creating near parent solutions and a small value of η allow distant solutions to be selected as offspring solutions.

$$\beta_i = \begin{cases} (2u_i)^{1/(\eta+1)} & \text{if } u_i \le 0.5 \\ \left[1/(2(1-u_i))\right]^{1/(\eta+1)} & \text{Otherwise} \end{cases} \tag{2.3}$$

After obtaining β_i using (2.3), offspring solutions are generated as follows

$$y_i^1 = 0.5((x_i^1 + x_i^2) + \beta_i (x_i^1 - x_i^2)) \quad y_i^2 = 0.5((x_i^1 + x_i^2) + \beta_i (x_i^2 - x_i^1)) \tag{2.4}$$

The SBX operator respects the interval schemata processing, in the sense that common interval schemata between parents are preserved in children. Thus during early iteration, parents being far away from each other, offspring are also created on the entire search space, thereby providing a good initial search of the entire space. When solutions converge near a good region, the parents are closer to each other and this operator provides a more focused search. This property makes the resulting GA self-adaptive.

2.2 Multi-parent Recombination operators

In this section, we see the details of multi-parent crossover operators. Usually, the crossover operator is applied to pairs of chromosomes, generating two offspring for each one of them, which are introduced in the population. However, multi-parent crossover operators have been proposed, which combine the features of more than two parents for generating the offspring. In general, sampling of more information from a population helps evolution process to bring better changes in the next generation. The studies on multi-parent recombination operators have given sufficient indication that they can enhance performance of GAs. The exploitative and explorative behavior of operators are due to use of probability distribution and more sampling of information due to use of more than two parents.

The multi-parent polynomial distribution recombination operator (MPX) [31] is a multi-parent extension of simulated binary crossover operator (SBX) [23] and the multi-parent lognormal distribution recombination operator (MLX) [31] is a multi-parent extension of SBX with lognormal distribution (SBX-l) [24]. Raghuwanshi et al. in [31] used MPX and MLX operators for single objective optimization. A brief description of the two operators is given below.

The prototype algorithm for the MPX operator is as follows:

a. From population select best parent and pick other (μ-1) solutions randomly.

b. For each gene (i=1,n) in real-parameter chromosome execute following steps

    i. Choose u_i randomly from the interval [0, 1].

    ii. Compute β_i using (2.5).

$$\beta_i = \begin{cases} (2u_i)^{1/(\eta+1)} & \text{if } u_i \le 0.5 \\ \left[1/(2(1-u_i))\right]^{1/(\eta+1)} & \text{Otherwise} \end{cases} \tag{2.5}$$

    iii. Calculate

$$D = \Big(\sum_{k=1}^{\mu} \Big(\sum_{j=1}^{\mu} | x_i^j - x_i^k | \Big)/\mu \Big)/\mu \tag{2.6}$$

    iv. Generate two genes around gene of best parent (say x^1) using (2.7)

$$y_i = x_{i1}^{1} \pm (\beta_i * D) \tag{2.7}$$

The operators based on polynomial distribution are more exploitative and exploitation range decreases with increase in distribution index of probability distribution (η) [31].

It is observed that operators with lognormal distribution are more explorative i.e it is capable to generate genes away from the parent gene. Their exploration range increases with increase in η. Also it is noticed that the probability of creating genes near the parent gene is almost zero [31].

A prototype algorithm for the MLX operator is as follows:

a. From population select best parent and pick other (μ-1) solutions randomly.

b. For each gene (i=1,n) in real-parameter chromosome execute following steps

   i. Choose u_i randomly from the interval [0, 1].

   ii. Compute β_i using (2.8).

$$\beta_i = \begin{cases} e^{-z\eta} & \text{if } u_i \leq 0.5 \\ e^{z\eta} & \text{Otherwise} \end{cases} \tag{2.8}$$

   Where z ~ N (0, 1) is standard normal variable.

   iii. Calculate D using (2.6)

   iv. Generate two genes around gene of best parent (say x^1) using (2.7)

Exploration and Exploitation

For multi-parent recombination operator, let’s consider (…,xij-1,xij,xij+1,…) ∈ [li, ui] genes to be combined to generate genes with αi<= xij and βi>= xij . The action interval [li, ui] can be divided into three regions: [li,αi], [αi, βi] and [βi ui]. These intervals may be classified as exploration or exploitation zones as shown in figure 2.1.

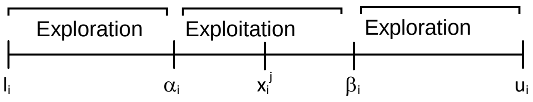

Fig 2.1 Action interval for **$(\ldots,x_i^{j-1},x_i^{j},x_i^{j+1},\ldots)$**

The exploration and/or exploitation degrees may be assigned to any recombination operator based on the way in which these intervals are considered to generate genes. The interval in which a generated genes lead to refinement is called as exploitation zone and genes generated in the interval leads to discover new search space is called exploration zone. During evolution process the values of α_i and β_i are changing.

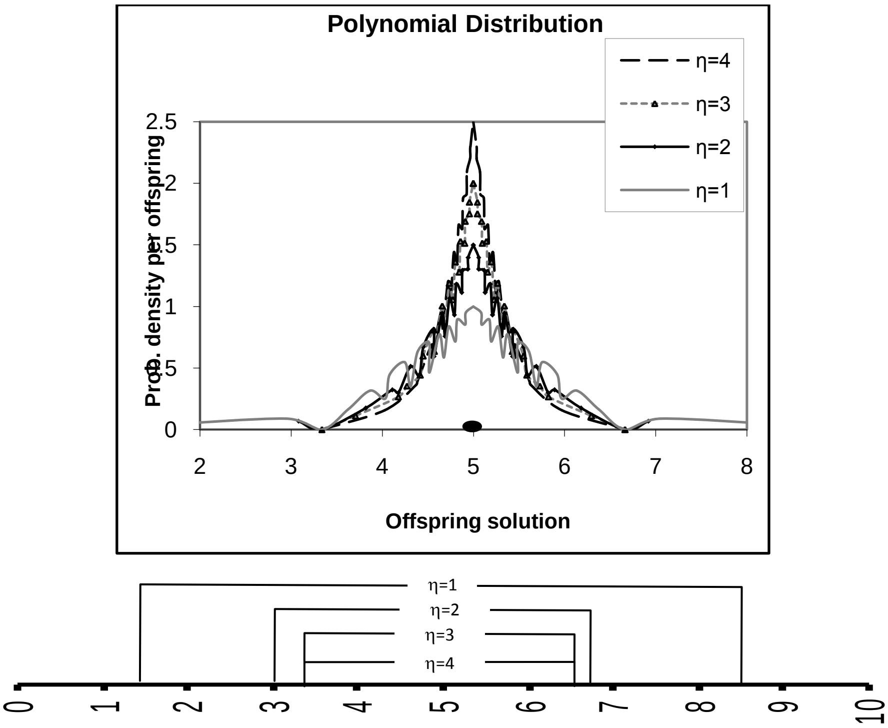

Fig 2.2 Distribution of offspring solution around parent solution at 5 for MPX

Fig 2.2 shows that the probability of generating offspring in the near neighborhood parent is more and the length of interval in which offspring are generated decreases with increase in η. The polynomial distribution based MPX operator is more exploitative and exploitation range decreases with increase in η.

Fig 2.3 shows that the probability of generating offspring away from the parent is more and the length of interval in which offspring are generated increases with increase in η. The lognormal distribution based MLX operator is more explorative and its exploration range increases with increase in η (shown by long tail in graph). For lognormal distribution probability of generating offspring near the parent is very less (shown by deep at 5 in graph) [31].

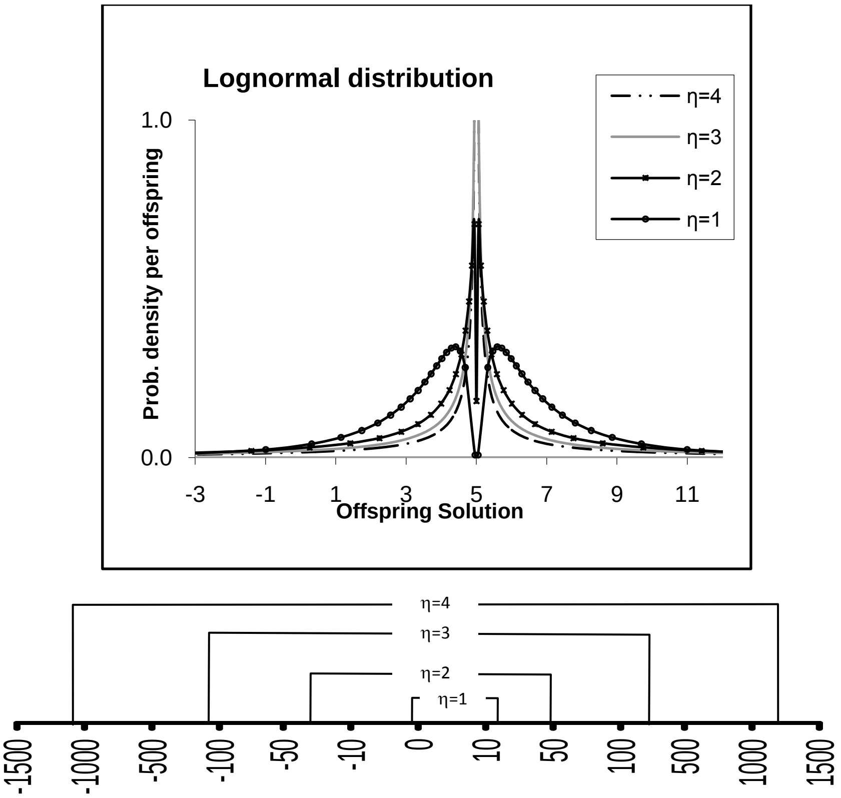

Fig 2.3 Distribution of offspring solution around parent solution at 5 for MLX

Any good search algorithm desires to explore a large search space in the beginning and the search should narrow down as it converges to the solution. If more than one parent is used in perturbation process, the range of perturbation may be adaptive and can be determined from the diversity of the parents on the fly. In self-adaptive recombination operators the extent of perturbation is controlled at run time. Operators like SBX, SBX-l, UNDX, and SPX have been tested for self-adaptive behavior. [31]

2.3 Modified NSGA-II

In section 2.1 a brief description of NSGA-II is presented. Many researchers around the world have accepted NSGA-II for its better diversity and faster convergence in solutions. However, the reviews of the NSGA-II suggest the following short comings of the algorithm:

In every generation NSGA-II performs nondominated sort on combined (parent + offspring) population.

Nondominated sorting is complex (O(MN^2)) and time consuming procedure.

Crowding distance check is done within the last front only. More diverse solutions may lie in other fronts also.

With the decrease of diversity among all solutions, search rates of overall solution lowers and local Pareto solutions is arrived prematurely because most of non-elitist solutions cannot participate in the genetic action fully, although the elitist strategy had improved the efficiency greatly.

The Pareto-domination based selection in NSGA-II aims at driving the whole population towards the PS (PF). However, it has no direct control over the movement of each individual in its population and then it has no good mechanism to control the distribution of its computational effort over different ranges of the PF or PS.

NSGA-II uses SBX crossover operator for recombination. SBX is parent-mean-centric crossover operator i.e. it produces offspring near the mean of two parents.

Above mentioned drawbacks of NSGA-II have opened scope for improvement and design some new schemes of selection of solution for formation of new generation to address the issues like convergence and diversity. Also there is need to design new MOGA frame work which is efficient. The crossover operator has always been regarded as the primary search operator in genetic algorithm (GA) because it exploits the available information from the population about the search space. Moreover, it is one of the components to consider for improving the behavior of the GA.

As mentioned in the previous section (section 2.2) MPX and MLX operator have given encouraging results on unimodal & multimodal single objective functions. In this study an attempt to improve performance of NSGA-II algorithm, MPX & MLX operators are deployed in NSGA-II framework by replacing SBX operator. These operators are gene-based parent centric crossover operator and more than two parent take part in recombination. The performance of modified NSGA-II is investigated on six multi-objective optimization problems.

2.4 Experimentation

An experimental study is performed to investigate the behavior of NSGA-II with multi-parent crossover operators MPX and MLX in terms of convergence and diversity. Modified real-coded NSGA-II is coded in MatLab 7.1. Table 2.1 shows parameter setting used for experimentation. Most of the parameters are same as given in [17]. Only few extra parameters are included for multi parent recombination operators. In this work NSGA-II will perform crossover for 90% times while go for mutation only for 10% times. As suggested in [31] we have kept number of parents involved in crossover operation as 5. Test problems used in the study are KUR, ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6. Table 1.2 in chapter 1 shows the number of objectives, number of variables, their bounds and the nature of the Pareto-optimal front for each problem.

TABLE 2.1 PARAMETER SETTING USED IN THE STUDY

Parameter	Values
GA type	NSGA-II
Population size (N)	100
Number of generations	250
Crossover probability parameter (p_c)	0.9
Probability Distribution indices for crossover η_c	2 - 8
Probability Distribution indices for mutation η_m	1 for MLX operator 20 for MPX operator
Probability of variable change (pvc)	0.5
Number of Parents	5
Number of children (λ)	2

Disruptiveness of chromosome depends upon the number of genes changed due to recombination operation. This disruptiveness on the one hand leads to more diverse exploration that can prevent premature convergence, but on the other hand, it slows convergence speed at the same time. For both the operators the value of crossover probability is moderated i.e. pvc=0.5.We use the polynomial mutation with mutation probability of $p_m = 0.1$ for real-coded NSGA-II.

TABLE 2.2 IGD METRIC FOR NSGA-II-MPX AND NSGA-II-MLX

Function	**NSGA-II-SBX**		**NSGA-II-SBX-l**		**NSGA-II-MPX**		**NSGA-II-MLX**	
	MeanIGD	**S.D.IGD**	**MeanIGD**	**S.D.IGD**	**MeanIGD**	**S.D.IGD**	**MeanIGD**	**S.D.IGD**
ZDT1	0.291342	0.040218	0.280034	0.030227	**0.278916**	**0.024618**	**0.27582**	**0.030171**
ZDT2	0.289128	0.045332	0.293881	0.059430	**0.273995**	**0.045896**	**0.264356**	**0.067330**
ZDT3	0.302956	0.028767	0.372282	0.049391	**0.274595**	**0.037762**	**0.265193**	**0.053949**
ZDT4	0.138006	0.034446	0.104872	0.039518	**0.087087**	**0.017339**	**0.055097**	**0.046780**
ZDT6	0.314778	0.059011	0.311945	0.093906	**0.266901**	**0.059899**	**0.235517**	**0.096936**
KUR	0.010036	0.001234	0.016382	0.010051	**0.001385**	**0.0114287**	**0.00113**	**0.00975**

2.5 Results and Discussion

30 random simulations are performed for each problem. All the results are taken after 250 generations. Six test problems are solved by NSGA-II-MPX and NSGA-II-MLX. Metric used for performance measure is Inverted Generational distance (IGD).Performance of NSGA-II-MPX & NSGA-II-MLX is compared with NSGA-II-SBX & NSGA-II-SBX-l. Table 2.2 shows the mean and variance of the IGD values for all six functions given by NSGA-II-SBX, NSGA-II-SBX-l, NSGA II-MPX and NSGA-II-MLX.

NSGA-II-MPX and NSGA-II-MLX have performed better than NSGA-II-SBX and NSGA-II-SBX-l in all functions. NSGA-II-MLX has better MeanIGD than NSGA-II-MPX but NSGA-II-MPX has less deviation. NSGA-II-MPX and NSGA-II-MLX are able to maintain a better spread of solutions in the obtained non-dominated fronts and have shown good convergence for multi-objective test problems. MPX and MLX are parent centric gene level crossover operators i.e. they create offspring near the best parent. If one parent among the selected parent is near to Pareto optimal front than MPX operator will create offspring near to this parent and hence better convergence will be achieved. SBX operators are Parent mean centric operators i.e. they create offspring near the mean of the two parents. If one parent is near to Pareto optimal front and other solution is away from Pareto optimal front, SBX operator will create offspring near the mean of two parents little away from Pareto optimal front. This is the reason SBX & SBX-l show poor convergence than MPX or MLX.

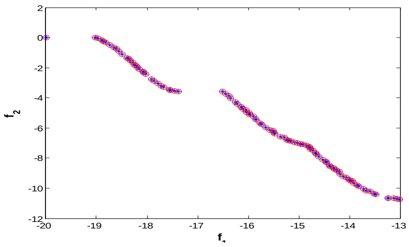

Fig 2.4. Nondominated Solutions with NSGA-II-MPX on KUR

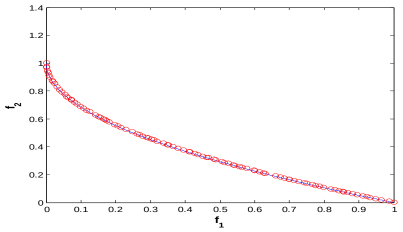

Fig 2.5. Nondominated Solutions with NSGA-II-MPX on ZDT1

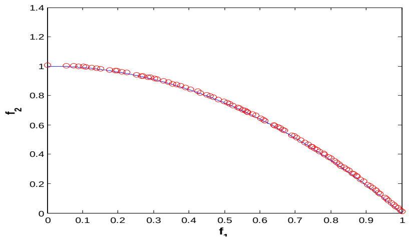

Fig 2.6. Nondominated Solutions with NSGA-II-MPX on ZDT2

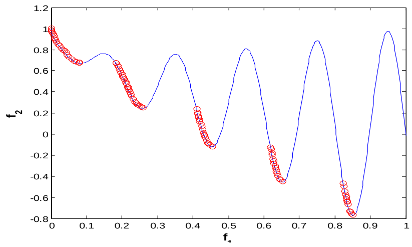

Fig 2.7. Nondominated Solutions with NSGA-II-MPX on ZDT3

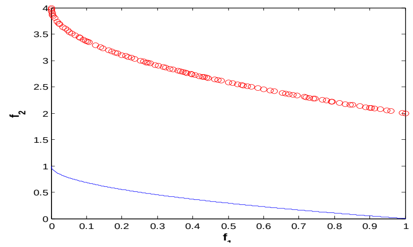

Fig 2.8. Nondominated Solutions with NSGA-II-MPX on ZDT4

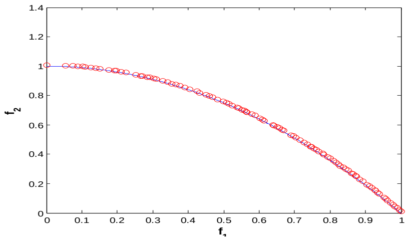

Fig 2.9. Nondominated Solutions with NSGA-II-MPX on ZDT6

Simulation results with MPX Operator: We have graphically shown the results obtained by NSGA-II-MPX with parameters p_c =0.9, pvc=0.5, η_c =1 and η_m =20 for problems KUR, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. These figures demonstrate the abilities of multi parent crossover operator to help in converging to the true front and in finding diverse solutions in the front. Figures given above shows all nondominated solutions obtained after 250 generations.

The problem KUR has three discontinuous regions in the Pareto-optimal front. Fig. 2.4 shows all nondominated solutions obtained after 250 generations with NSGA-II-MPX. For ZDT1 function, the Pareto-optimal front shown in fig 2.5 is continuous and has a uniform distribution of solutions across the front. The nondominated solutions on the problem ZDT2 is shown in Fig 2.6. This problem has a non-convex Pareto-optimal front. NSGA-II-MPX has found a better spread and more solutions in the entire Pareto-optimal region. Fig 2.7 shows nondominated solutions for ZDT3 function. The Pareto-optimal front is discontinuous and NSGA-II-MPX has got success in finding all the discontinuous regions with uniform spread of non-dominated solutions. Real-coded NSGA-II-MPX get stuck at local Pareto-optimal front as shown in fig 2.8, but the convergence and ability to find a diverse set of solutions are definitely better for ZDT4. The problem ZDT4 has 21^9 or $7.94(10^{11})$ different local Pareto-optimal fronts in the search space, of which only one corresponds to the global Pareto-optimal front. The Euclidean distance in the decision space between solutions of two consecutive local Pareto-optimal sets is 0.25. For ZDT6 problem, the non-convex Pareto-optimal front is shown in fig 2.9. The density of non-dominated solution is thick towards Pareto-optimal front. The exploitative nature of MPX operator has produced a search-bias that has helped the algorithm to converge better in all problems.

Simulation results with MLX Operator: Simulation results of NSGA-II-MLX for six different test problems i.e. KUR, ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 with parameters p_c =0.9, pvc=0.5, η_c =1 and η_m =1 are shown in fig 2.10-2.15.

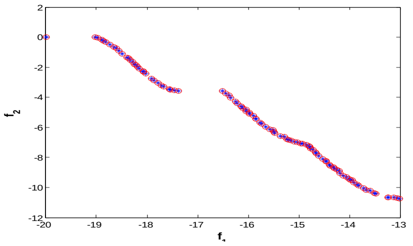

Fig 2.10 Nondominated Solutions with NSGA-II-MLX on KUR

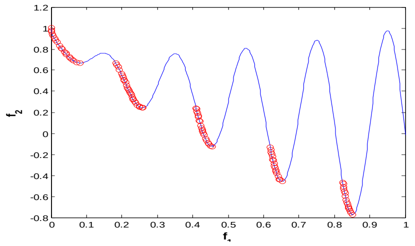

Fig 2.13 Nondominated Solutions with NSGA-II-MLX on ZDT3

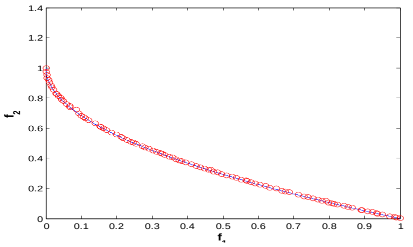

Fig 2.11 Nondominated Solutions with NSGA-II-MLX on ZDT1

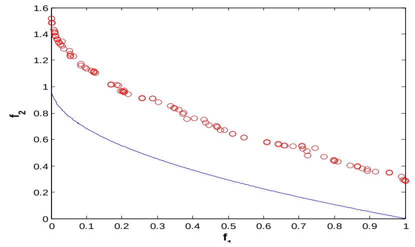

Fig 2.14 Nondominated Solutions with NSGA-II-MLX on ZDT4

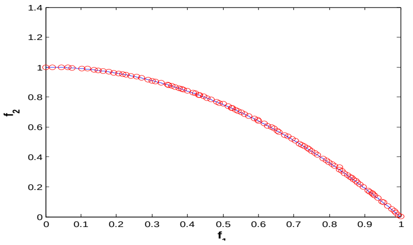

Fig 2.12 Nondominated Solutions with NSGA-II-MLX on ZDT2

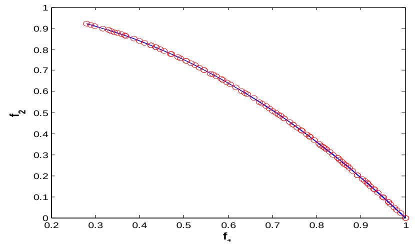

Fig 2.15 Nondominated Solutions with NSGA-II-MLX on ZDT6

The algorithm has converged to Pareto-optimal front for functions like ZDT1, ZDT2, ZDT3 and ZDT6 in 250 generations. Function ZDT4 requires more generations to converge to Pareto-optimal front. Around 400 generations are required for convergence near to Pareto-optimal front. MLX operator has given excellent performance for ZDT6 problem. Fig 2.15 shows Pareto-optimal front, which is non-uniform with very good density of non-dominated solutions on the Pareto optimal front. Lognormal distribution is explorative in nature hence MLX operator produces offspring away from parent. Also it samples more than two parent, more diverse solutions it produces which helps in a better search of decision variable space. NSGA-II-MLX has given good performance with all the six problems.

Effect of Probability Distribution Index for crossover η_c : We have tested MPX and MLX operators for parameter probability of distribution index η_c . For MPX operator we have selected parameter p_c =0.9, pvc=0.5, η_m =20 and η_c = 2 to 8 in step of 2. Fig 2.16-2.19 Shows simulation results for MPX operator for ZDT1 problem. It is observed that with increase in the value of probability distribution index for MPX crossover η_c , the Pareto-optimal front deteriorates i.e. discontinues with less number of non-dominated solutions in Pareto front and convergence is also weak.. The reason is the exploitation capability of MPX that decreases with the increase η_c Hence the study suggests keeping value of η_c in the range of 1-4.

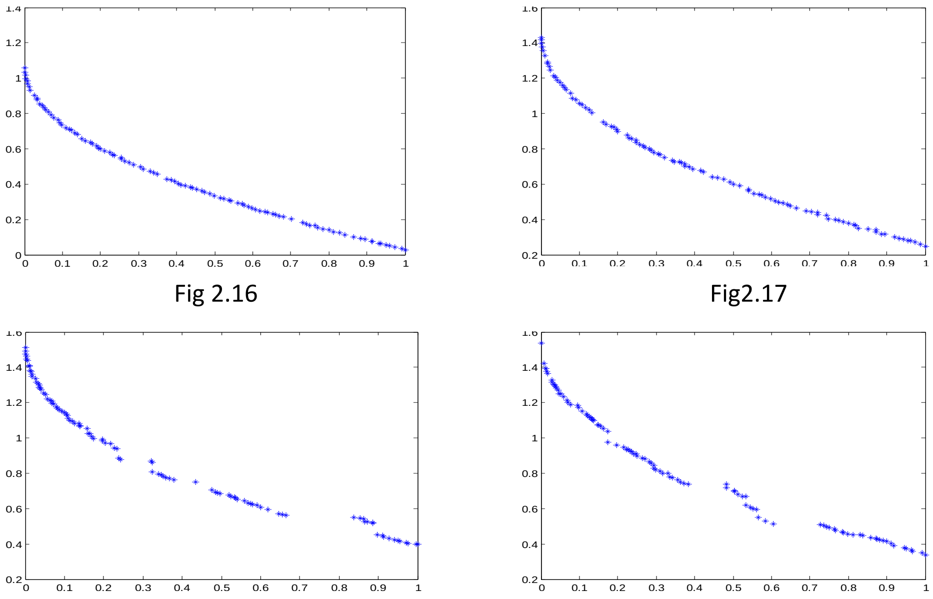

Fig 2.16

Fig2.17

Fig 2.18

Fig 2.19

Fig 2.16-2.19 show effect of different values of distribution index for MPX crossover on Pareto-optimal front and non-dominated solutions. Values of η_c for fig 2.13-2.16 are 2, 4, 6 and 8 respectively

MLX operator with parameter p_c =0.9, pvc=0.5, η_m =1 and η_c = 2 to 5 in step of 1 is tested on ZDT1 function. Algorithms behavior is shown graphically in fig 2.20-2.23.

ZDT1 is a simple problem and the algorithm convergences very nicely with MLX operator. Since MLX is explorative operator and it exploration (capability to produce genes away from parent genes) increases with increase in probability distribution index η_c. Higher explorative power has produces gene very much away from the parent genes and hence there are less number of non-dominated solution in the Pareto front. Poor performance of NSGA-II-MLX with higher values of η_c is seen in fig 2.20-2.23. Therefore we suggest keeping value of η_c 1 or 2.

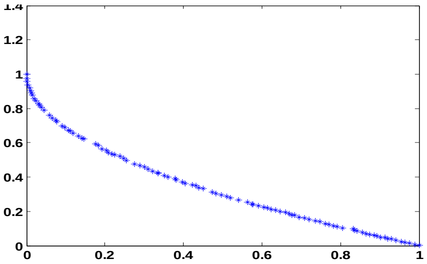

Fig 2.20

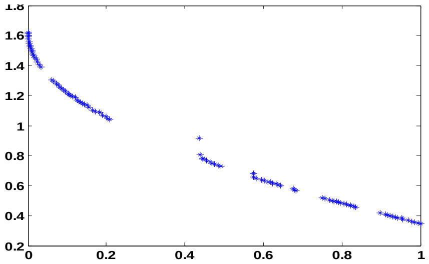

Fig 2.21

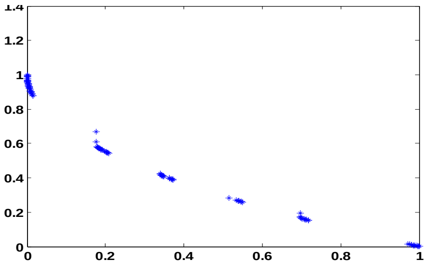

Fig 2.22

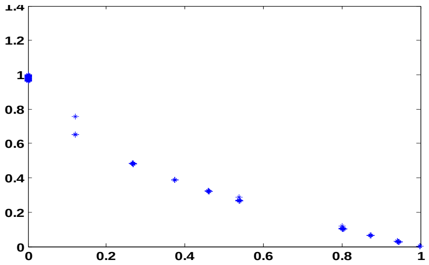

Fig 2.23

Fig 2.20-2.23 show effect of different values of distribution index for MLX crossover on Pareto-optimal front and non-dominated solutions.

Values of η_c for fig 2.17-2.20 are 2, 3, 4, and 5 respectively

2.6 Key findings

This chapter presents study of NSGA-II algorithm and some drawbacks of the schemes used in the algorithms for convergence and diversity preservation among the solutions. These observations points towards new research directions. Multi-parent recombination operators, MPX & MLX have been described. NSGA-II is modified by replacing SBX operator with MPX & MLX operators. This is an attempt to investigate suitability of MPX & MLX operator for multi-objective optimization. Performance of NSGA-II-MPX & NSGA-II-MLX are tested on six test problems, KUR, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6.

Following conclusions are drawn from this study,

The empirical result shows that, the use of multiple parent recombination operators, MPX & MLX, have produced good results.

Use of more than two parent solutions helps operator to sample more features into offspring that accelerated speed of convergence to global Pareto optimal front.

NSGA-II-MPX and NSGA-II-MLX are able to maintain a better spread of solutions in the obtained nondominated fronts and shown good converge for multi-objective test problems. MPX and MLX operators have improved performance of NSGA-II algorithm.

Exploitation and exploration capability of MPX & MLX operator depends on value of probability distribution index. This study suggest range of probability distribution index as [1, 4] for MPX operator and probability distribution index as 1 or 2 for MLX operator.

Chapter 3

Rank-based Selection Methods for MOGA

In multi-objective Genetic algorithm a selection of individual is the crucial issue because the algorithm has to achieve two goals i.e. to maintain diversity along the Pareto optimal front and convergence towards the Pareto optimal front. The selection methods used for formation of next generation can be split broadly into Pareto and non-Pareto based selection methods. Non-Pareto based selection methods are also called as rank-based selection methods. This chapter discusses rank-based selection methods. Rank sum based Multi-objective Genetic Algorithm (RSMOGA) and Summation of Normalized Objective Value based Multi-objective Genetic Algorithm (SNOVMOGA) are introduced and their performance is investigated in terms of convergence and diversity on MOPs. This chapter also speaks about comparison of rank based selection schemes with Pareto dominance based selection schemes.

3.1 Introduction

Many researchers have shown that Genetic Algorithm (GA) is a robust search and optimization tool and has got the potential to solve optimization problems. Real-world optimization problems have more than one objective normally in conflict with each other. Due to the existence of multiple objectives, the selection criterion used for single objective is not suitable for multi-objective in evolutionary algorithms (MOEAs). The superiority (or dominance) of one solution over the other cannot be established with many objectives in mind. In multi-objective Genetic algorithm a selection of individual is the crucial issue because the algorithm has to achieve two goals i.e. to maintain diversity along the Pareto optimal front and convergence towards the Pareto optimal front. The selection process in multi-objective GA occurs at two different phases

1. Selections of individuals for reproduction i.e. chosen individuals those undergo crossover and mutation operation.
2. Selection of individuals for formation of a next generation population.

Selection mechanism should select individuals which are close to Pareto front and are well separated. Selection mechanism is concerned with convergence towards Pareto-optimal front and maintenance of diversity among the selected individuals [1]. Most of the selection mechanism used for reproduction is based on the fitness of individuals in the population. The fitness of a solution depends upon how well the solution addresses the objective function of the target problem. Selection usually favors the fitter individuals. The selection pressure can be control by varying the level of emphasis the process assigns to fitter individuals. A more stringent selection process that is more biased towards the fitter solutions will lead to exploitation while a less stringent selection process will put less pressure lead to exploration of population. The state-of-the art methods used for selection of parents for reproduction are proportionate selection, tournament selection and Rank-based selection. In proportionate selection (Roulette Wheel Selection or Spinning Wheel Selection) each individual is assigned a selection probability based on its fitness over the total fitness of the entire population. Once this is done, individuals are concatenated with each other to form a spinning wheel, with each portion of the wheel representing an individual and the size of each portion representing the corresponding selection probabilities. The basic idea of tournament selection scheme is quite straightforward. A group of individuals is selected randomly from the population. The individuals in this group are then compared with each other, with the fittest among the group becoming the selected individual. Rank-based selection schemes first sort individuals in the population according to certain criteria (usually according to their fitness). A function is then used to map the indices of individuals in the sorted list to their selection probabilities. All these methods can be applied directly to single objective problem but for multi-objective problem they need a single scalar fitness assignment method. [1]

The selection methods used for formation of next generation can be split broadly into Pareto and non-Pareto based methods. Pareto based methods (MOGA, NSGA-II, SPEA etc.)[16, 17, 61] rely on ranking the population based on the direct measure of Pareto dominance within the population. A non-domination sorting process is executed in each of the generation to select the fitter solutions. The steps of finding the nondominated solutions are complicated and time consuming. Beside this, if the selection of parents is only based on the nondominated front number, the diversity of the final solution is likely to be reduced in some problems. For the commonly used non-domination sorting, the complexity to obtain the overall non-dominated set is $O(MN^2)$.

The non-Pareto based methods (for eg. Vector Evaluated GA) [35] generate a Pareto set implicitly, without making a direct comparison to check domination/non-domination with other members of the population. There are various non-Pareto based methods called as Rank-based method. Rank-based selection schemes first assign single scalar fitness to each individual and then sort individuals in the population according to single scalar fitness. Then a function is used to map the indices of individuals in the sorted list to their selection probabilities. Although this mapping function can be linear (linear ranking) or a non-linear (non-linear ranking) but the idea of rank-based selection remains unchanged. However, the performance of a selection scheme depends largely on this mapping function.

Rank-based selection methods use ranking technique to assign ranks to solutions. Ranking technique used should be Range-independent (effective ranges of the objective functions are the same). Range-independent ranking techniques require no knowledge of the effective range of each objective function that makes them independent of the nature of the objectives and overall problem itself. In addition to being range-independent, there is another significant, and usually overlooked property that, a good ranking method should have the ability to increase the 'importance' of some objectives with respect to others in ranking of solutions so as to allow search to be directed to converge on Pareto-optimal front. *Importance* is a simple way to give ranking method additional problem-specific information, in order to direct a GA to faster convergence. [64]

In this work we propose to use Rank-sum (RS) based technique as discrimination (ranking) technique along with adaptive-diversified selection method for formation of next generation. Rank-sum based Technique finds ranks of solutions in terms of each separate objective and use these ranks with a suitable aggregation function to calculate fitness values (Rank-sum) for the solutions. Adaptive-diversified selection mechanism use Rank-sum for formation of next generation of population of multi-objective genetic algorithm.

In order to be effective, the Rank-based method must provide a fine-grained discrimination by considering how significantly better is a solution from the others with respect to each objective. Discarding this information can lead to wrong discrimination decisions and, can thus; negatively affect the search capabilities of a MOEA. To achieve this we propose Summation of normalized objective value (SNOV) ranking (discrimination) techniques along with improved selection method. Summation of normalized objective value based Technique normalizes each objective value and calculates sum of these normalized values of all objectives of an individual and uses this sum to assign ranks to the individual. Along with SNOV procedure, improved diversified selection with a pre-selection mechanism is used to overcome the problems that are encountered in Rank-sum based method.

3.1.1 Related work

High computational complexity associated with dominance based selection schemes has opened a scope for the development of rank-based (non-Pareto) selection schemes, which are less computationally complex. Bentley and Wakefield [64] proposed a preference ordering over a set of nondominated points for 2-objective problems. They compared six such ranking methods. They designed four methods: 1) *weighted average ranking* (WAR), 2) *weighted maximum ranking* (WMR – a basic extension to VEGA [35]), 3) *sum of weighted ratios* (SWR), and 4) *sum of weighted global ratios* (SWGR). The other two were, non-dominated sorting, and the equivalent of single-objective fitness (i.e. summing the objective values). In their methods any point in a collection *P* (whether nondominated or not) is considered as a vector of ranks. The sum of the elements in this vector provides a way to rank the points. All six methods were used with a basic binary coded genetic algorithm.

Drechsler et al [65], proposed and tested the *favor* relation technique to provided a finer grained ordering over multi-objective points Essentially, point *s* is favored over point *t* if *s* is better than *t* on more objectives than on which *t* is better than *s*, and if we treat the pair wise favor relations as edges in a graph (and do some necessary processing), we can obtain an ordering. Di Pierro et al in [66, 67] have also investigated preference ordered ranking for multi-objective points. The technique offered the notion of 'efficiency of order *k*', or '*k*-optimality'. Another method *winning score* is proposed in [68]. The idea of the 'Compressed Objective Genetic Algorithm' (COGA) [68] is to treat a many objective problem as a 2-objective one, where one objective is *winning score*, which can impose an ordering on nondominated points, and the other is a helper-objective that ensures diversity. In [69] Knowles et al evaluated the relative quality of various ranking methods for nondominated-point and found that the average ranking (ARF) method is highly effective in comparison with the other methods. The next-best method was *k*-optimality, while favor and random selection from the Pareto front both tended to do well. Knowles et al have also proposed metric to assess rank distributions in terms of their *relative entropy*. If relative entropy is close to 1 it is 'ideal' situation in which the points are totally ordered over |*N*| distinct ranks. It is zero in the case where the distribution simply gives every point the same rank.

Suganthan et al. [70] proposed Rank-sum based sorting that works well for multi-objective evolutionary programming (MOEP). This is very simple and effective ranking technique for selection of nondominated solutions. Suganthan et al. have also proposed fast sorting technique and improved selection scheme for multi-objective differential evaluation (MODE) in [71] [72].

All the above-mentioned ranking schemes are simple schemes of scalar fitness assignment in multi-objective problems. Schemes given in [64-69] are tested on relatively simple multi-objective test problems whereas schemes given in [70-73] are tested on complex problems. In this work use the schemes given in [70 & 71] in a new Multi-Objective Genetic Algorithms (MOGA) framework.

3.2 Two Rank-based selection methods

In this section we review Rank-sum based selection and Summation of Normalized Objective Value based selection methods in detail. In this study we propose following two MOGA that employ rank-based selection methods

RSMOGA (Rank-sum Sort based Multi-objective GA) uses Rank-sum (RS) ranking (fitness assignment) method along with adaptive diversified selection mechanism for multi-objective genetic algorithm.

SNOVMOGA (Summation of Normalized Objective Value based Multi-objective Genetic Algorithm) uses Summation of Normalized Objective Values (SNOV) based ranking technique and improved diversified selection mechanism as discrimination technique for selection of individuals of next generation in a multi-objective genetic algorithm.

3.2.1 Rank-sum Sorting

Rank-sum sorting is a simple discrimination technique and can be used for making discrimination among the solutions in presence of multiple objectives. It is a range-independent ranking method i.e. fitness ranking of solutions, defined by the ranking method *does not* change when the effective range of objective changes. The idea is to divide every objective's range into 10 ranks for a population of 10 solutions. Then assign rank to individual objective-value of solution and calculate sum of all the objective ranks of the solution. This single scalar value is called Rank-sum (fitness) of the solution. An Algorithm for Rank-sum calculation is as follows. We assumed that all are minimizing problems (maximizing problems can be converted to minimizing problems by multiplying -1):

Step 1 Select one unranked objective

Step 2 Get the maximum and minimum value of the selected objective to calculate the range for this objective

Step 3 Divide the objective’s range into 10 grids (10 fuzzy ranks) i.e. grid size = (maximum-minimum) / Number of Ranks

Step 4 For every point in the search space, identify which grid it belongs to and assign the corresponding rank to the point for the selected objective.

Step 5 If all objectives have been selected go to step 6, otherwise repeat Step 1-4.

Step 6 Sum the rank of all objectives of each solution to obtain the Rank-sum of the solution. Also obtain the Rank-sum for all solutions in population.

Example: For a population of ten solutions, if minimum objective value = 0.1 and maximum objective value = 1.0 then grid size will be 0.09. Solution having Minimum objective value is assigned Rank 1 and solution having maximum objective value is assigned rank 10. Ranks of remaining solutions having objective value (x) is decided as

0.1>x<= (0.1+0.09=0.19) then Rank is 2

0.19>x<= (0.1+2*0.09=0.28) then Rank is 3

…

0.63>x<= (0.1+8*0.09=0.82) then Rank is 9.

Working of Rank-sum calculation algorithm is illustrated with the sample example as shown in Table 3.1.

TABLE 3.1 RANK-SUM ASSIGNMENT TO 10 SOLUTIONS

Objective1	**Rank1**	**Objective2**	**Rank2**	**Rank1+ Rank2 = Rank-sum**
0.431	4	0.340	4	8
0.314	3	0.364	4	7
0.393	4	0.591	7	11
0.934	9	0.264	3	12
0.314	3	0.364	4	7
0.393	3	0.591	7	10
0.100	1	0.038	1	2
0.458	5	0.869	10	15
1.000	10	0.664	8	18
0.870	8	0.012	1	9

It is an alternative to computationally complex non-domination sorting. Rank-sum calculation algorithm Step 2 requires O(N) comparisons to find the maximum and minimum value. Step 4 requires O(N) comparisons to identify the corresponding rank. Step 5 recursively call step 1-4, thereby requiring O(MN) (where M is the number of objectives and N is the number of solutions) computations of the above procedure. For non-domination sorting, the complexity to obtain the overall nondominated set is O (MN^2). Complexity of Rank-sum is linear and hence it requires less CPU time. All these observations lead us to select Rank-sum sorting for multi-objective genetic algorithm.

3.2.2 Adaptive Diversified selection (DS) scheme

Selection scheme used in the formation of population for next generation in any genetic algorithm is responsible for maintaining the diversity i.e. solutions should be distributed in such a way that they cover entire Pareto front. A scan percentage P is set and all the individuals having Rank greater than P will not be considered for selection. The proposed diversified selection scheme adaptively set this scan percentage. In the initial generation P is set to 90 to 80 of total rank and as generation progresses P is gradually reduced to 50 percent of the total rank. The adaptive diversified selection scheme maintains two sets of population namely, preferential set and backup set. All the solutions with rank less than P and minimum Rank-sum are placed in the preferential set where as solution with higher Rank-sum will be kept in back-up set. Solutions from preference set will be selected as members of next generation. If there is insufficient number of solutions in preference set then solutions from back-up set will be selected as members of next generation. The steps of building the preferential set are as follow:

Step 1 Select one unselected objective

Step 2 For the selected objective, scan P percentage of the total ranks. For each rank (if this rank is not empty, otherwise just continue to the next rank) of the selected objective, the solutions with the lowest Rank-sum will be chosen to enter preferential set.

Step 3 If all objectives have been selected go to step 4. Otherwise repeat Step 1-2.

Step 4 Collect the solutions not inside the preferential set and put them in the backup set.

Table 3.2 shows an example of formation of preferential set. First select solution having minimum objective1 rank (obj1_rank=1 i.e. solution7), check if other solutions have same obj1_rank. No other solutions have same obj1_rank. Place, solution7 in the preferential set. Now select next minimum obj1_rank (obj1_rank=3). Since more than one solutions have obj1_rank=3 select solution having minimum Rank-sum. But solution 2 and 5 have same Rank-sum (Rank-sum=7). In this situation selection scheme will select randomly between 2 and 5. Here solution 2 is chosen and placed in preferential set. Repeat the procedure for all the obj1_rank less than P (P=9). Thus we get preferential set according to objective 1 having solutions 7,2,1,8,10. Now select objective 2 and repeat the same procedure. We get preferential set according to objective2 having solutions 7,4,2,6,9. Combine the two sets and remove duplicates. The final preferential set has solutions 1,2,4,6,7,8,9 and 10. Solution 3 and 5 will be placed in backup set.

TABLE 3.2 RANK-SUM BASED SELECTION METHOD

S No	Obj1 rank	Obj2 rank	Rank sum	Sol^n in Pref set (according to obj1)	Sol^n in Pref set (according to obj2)	Sol^n in Pref set
1	4	4	8	7,2,1,8,10	7,4,2,6, 9	1,2,4, 6,7, 8,9 and 10
2	3	4	7			
3	4	7	11			
4	9	3	12			
5	3	4	7			
6	3	7	10			
7	1	1	2			
8	5	10	15			
9	10	8	18			
10	8	1	9			

3.2.3 Ranking based on SNOV

Ranking method must provide a fine-grained discrimination by considering how significantly better is a solution from the others with respect to each objective. Suganthan et al. proposed the SNOV (Summation of Normalized Objective Value) based ranking technique for differential evaluation [72]. We have planned to use it in Genetic algorithm.

Throughout the GA evolution, every separate objective (fitness) function in a multi-objective problem will return values within a particular range. This 'effective range' of every objective function is determined not only by the function itself, but also by the domain of input values produced by the GA during evolution. These values are the parameters to be evolved by the GA and their exact values are normally determined initially by random, and subsequently by evolution. Every separate objective function will have a different effective range. This means that a bad value for one could be a reasonable or even good value for another. If the results from these two objective functions were simply added to produce a single fitness value for the GA, the function with the largest range would dominate evolution. GA should treat all objectives in a multi-objective problem should be treated equally i.e. all the effective ranges of the objective functions should be the same. To make effective range of the entire objective functions equal, we normalize objective values. After normalization effective range of all the objective function will be zero to one. The summation of normalized objective value will be treated as single scalar fitness of the solution. The SNOV procedure is given below:

Step 1 For m = 1 to M (number of objectives)

Step 2 Find the maximum max $_f$ and minimum min $_f$ objective values of the m^th^ objective and calculate the range of the m^th^ objective.

Step 3 Normalize the m^{th} objective values of all members using the equation below: $f_m(x)=(f_m(x) - f_{min})/ (f_{max} - f_{min})$

where f_m is the normalized m^{th} objective value.

Step 4 Endfor

Step 5 For i = 1 to NP (population size)

Step 6 Sum all normalized objective values of the member to obtain a single value.

Step 7 End for

Improved selection (IS)

Diversified selection proposed in [70], maintains diversity but the problem with the scheme is excessive selection of individuals along the two objective axes i.e. more number of individuals having minimum value of objective are selected. To achieve diversity along the front it is expected that more number of solutions having tradeoff of objectives should be selected in the approximate set of individuals. A genetic drift is seen in the population as generation progresses. To overcome this problem, an improved diversified selection is proposed. In the proposed method, a pre-selection is added before the diversified selection. In the pre-selection, a reference point is identified in every generation and this point is used to remove the bad individuals in the current population. The reference point (in normalized objective space) is identified using the following equation:

$$objective = 0.5 - (0.5 * \text{gen} / \text{Max\_gen}) \qquad (3.1)$$

where o*bjective* is the i^{th} objective value of the reference point (all the objective value are the same). *Max_gen* is the maximum generation number while *gen* is the current generation number. The reference point is starting at the centre of the objective space and gradually moves to the origin along the search process. Figure 3.1 (line) shows how the reference point moves in a two objective case. With the reference point, some individuals are removed from the current population before diversified selection. The details are given below

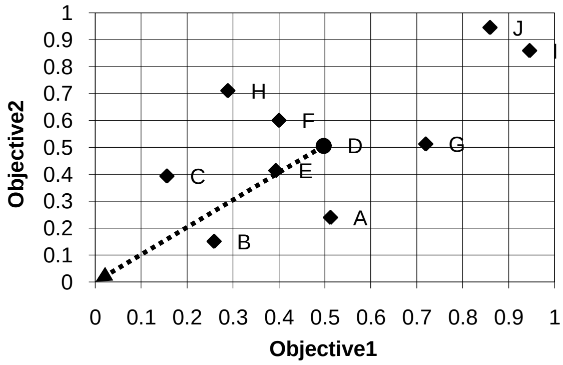

Fig.3.1. Reference point selection

Step1 Identify the reference point using equation (3.1)

Step2 Find the closest individual in population to the reference point and set it as reference individual.

Step3 Remove all the individuals that are dominated by the reference individual.

All the selected solutions are kept in the preferential set and remaining solutions are kept in backup set. The solutions inside the preferential set will be selected first for evolving. If there are insufficient solutions in the preferential set, solutions in the backup set will be selected based on the summation of normalized objective values. The steps for obtaining the two sets are given as:

Step1 For m = 1 to M (number of objectives)

Step2 Evenly divide the range of the objective space into 100 bins.

Step3 Scan P percentage of the 100 bins (i.e. from bin 1 to P, P may be chosen as 80 or 90).

Step4 For each scanned bin (if this bin is empty, otherwise just continue to next bin), the solution with the smallest summation of normalized objective values will be chosen to enter preferential set.

Step5 End For

Step6 Accumulate the solutions excluded from the preferential set and store them in backup set.

An example of above-mentioned techniques is given in the table 3.3. The two objective values are normalized using the SNOV procedure. Table 3.3 shows the normalized objective values of population of size 10 for the first generation. Divide objective space into 100 bins. The reference point is at the centre of the objective space whereas the reference individual is point D. All the individuals that are dominated by point D (i.e. points G, I and J) will be removed from current population and are kept in backup set. Now set the bin scan percentage P to 80 and start selecting individuals from current population for preferential set. First bin is scanned and individual present in the bin will be kept in Preferential set. If two or more individuals belong to same bin than individual having lower SNOV value will be kept in preferential set and other individuals will be kept in backup set. Next bin will be scanned and individuals will be selected for preferential set. In this way 80% bins will be scanned and A, B, C, D, E, F and H are kept in preferential set. Note that at least NP individuals should be kept as parents for next generation. If too many individuals are removed, the ones with better SNOV will restored until the number of individuals reach NP (In this case, the diversified selection step is skipped). Complexity of scanning procedure is (M*Number of Bins Scanned*N) where N is number of individuals in current population.

TABLE 3.3 SUMMATION OF NORMALIZED OBJECTIVE VALUES AND IMPROVED SELECTION OF INDIVIDUALS FOR FIRST GENERATION OF SNOVMOGA

Pop	Normalized objective value1	Normalized objective value2	Summation of Normalized objective values	Ref. Pt.	Pref. set
A	0.51231	0.00011	0.51242	**D**	**A,B, C, D, E, F and H**
B	0.25901	0.15032	0.40933		
C	0.00032	0.39321	0.39353		
D	**0.49897**	**0.50708**	**1.00605**		
E	0.39321	0.41367	0.80688		
F	0.40027	0.60011	1.00038		
G	0.71967	0.51231	1.23198		
H	0.28901	0.71067	0.99968		
I	1.00000	0.85966	1.85966		
J	0.85945	1.00000	1.85945		

3.3 Experimental design

An experimental study is performed to investigate the behavior of the selection approaches in terms of convergence and diversity. To demonstrate the proposed schemes of ranking for selection of parents in MOGA, we have implemented MOGA using MatLab 7.1. The experimentation carried out for two different MOGAs:

Part A: RSMOGA uses RS ranking method along with adaptive diversified selection mechanism for multi-objective genetic algorithm.

Part B: SNOVMOGA uses SNOV based ranking technique and improved diversified selection mechanism

Common settings and details of experimentation for Part A and Part B are given below:

GA procedure: The MOGA procedure used for experimentation is given below. Part A uses it with RS-DS and Part B uses it for SNOV-IS.

Step1. Initialize Population (P) of size N

Step2. Evaluate objective functions and assign fitness to each individual of P

Step3. Apply RS/SNOV procedure on P to assign scalar fitness to every individual in P

Step4. Select individuals from P for crossover using Multi-level Tournament selection.

Step5. Perform crossover in initial generations using MLX operator [31] and in later generations algorithm uses MPX operator [31].

Step6. Repeat Step 4 and 5 till Offspring Population (OP) size = N

Step7. Combine P and OP to produce Combined Population(CP) of size=2N. Remove all individuals from P and OP.

Step8. Apply Ranking scheme (RS/ SNOV) on CP

Step9. Select N individuals as parent for next generation using adaptive diversified selection / improved selection method and store them in P

Step10. Check, if the stopping criterion (Max. No. Of generations) is met, and then present individuals in P as Pareto optimal solutions. Otherwise, the procedure is repeated from Step 2.

Any good search algorithm must explore a large search space in the beginning and the search should then narrow down as it converges to the solution. To support this property explorative operator MLX is used in the initial generations (30-40 percent of maximum generation) and in the later generations exploitative operator MPX is used in the algorithm.

Since algorithm is using parent centric multi-parent crossover operators a multilevel tournament selection is used. The steps for multilevel tournament selection are given below:

Select two pairs of solutions randomly. Compare Rank-sum/SNOV of each pair. The comparison of two pairs will generate two winner solutions. Again Compare Rank-sum/SNOV of pair winner solutions. This will produce the Tournament winner solution. This tournament winner is treated as best parent and MLX & MPX generate offspring near this best parent.

Multilevel tournament selection increases selection pressure that give better chance to select more fit parents for reproduction. That may help to speed up the convergence.

Multi-parent crossover operators: Multi-parent recombination operator combines the features of more than two parents to generate offspring. In general, sampling of more information from a population helps evolution process to bring better changes in the next generation. The performance of RCGA on a particular problem strongly determined by the degree of exploration and exploitation associated with the crossover operator being applied. MPX and MLX are the two parent centric multi-parent recombination operators where child solutions are generated around one parent solution using other parent solutions. The degree of exploration and exploitation can be controlled by using a proper value of distribution index of probability distribution used by them. MPX operator is more exploitative in nature and the exploitation range decreases with increase in η. The MLX is explorative in nature i.e. it is capable to generate genes away from the parent gene. Its exploration range increases with increase in η. [17]

Test problems and performance indicator: The test problems (UF1-UF9) in this work are taken from CEC2009 special session and competition [10]. The IGD (Inverted Generational Distance) metric is used as performance indicator to quantify the quality of the obtained results [1]. The IGD metric measures "how well is the Pareto-optimal front represented by the obtained solution set".

PC configuration:

System: Microsoft Windows XP

RAM: 2.00 GB

CPU: Pentium 4, 2.99GHz

Parameter settings:

Population size (N): 100 (for 2 objectives) and 150 (for 3 objectives)

Number of generations: 3000

Crossover probability (pc): 0.9

Probability Distribution indices for MPX : 2

Probability Distribution indices for MLX : 6

Number of Parents: 5

Number of children (λ): 2

Bin scan Percentage P(Improved selection) : 80%

Scan Percentage P (Adaptive diversified selection): 90%-50%(P reduces by 10% after every (Number of Generations/10) generations till P reduces to 50%).

TABLE 3.4 THE IGD METRIC FOR RSMOGA

Problem	**Min**	**Max**	**Mean**	**Std**
UF1	0.042128	1.000722	0.064189	0.095339
UF2	0.014962	0.546735	0.020020	0.016462
UF3	0.022385	0.044631	0.031159	0.004444
UF4	0.012623	0.059979	0.028890	0.006118
UF5	0.774876	3.776223	0.978930	0.407084
UF6	0.265128	2.838803	0.370197	0.289605

UF7	0.010607	0.669267	0.019931	0.013800
UF8	1.225456	4.956260	1.83609	0.523283
UF9	0.550592	5.966486	0.988528	0.580680

3.4 Discussion on results

This section covers discussion on results obtained in experimentation

Part A: 30 random simulations are performed for each problem with RSMOGA The minimum (Min), maximum (Max), mean, and standard deviation (Std) of the IGD metric are reported in Table 3.4 and figure 3.2-3.10 shows plot of approximate solutions for function UF1-UF9 with RSMOGA. For seven bi-objective instances, RSMOGA found good approximation to UF3, UF4 and UF7 but performed poorly on UF5 and UF6. It has given comparable results for UF1 and UF2.For three objective instances; RSMOGA has shown very poor performance for UF8 and UF9.

RSMOGA algorithm converges to near to Pareto-optimal front with good spread in nondominated solutions on Pareto-optimal front in less than 1000 generations for some functions. Further use of diversified selection of parents for next generation has contributed to good spread of solutions as evident from the IGD values given in the table V. For other functions to some extent it has found global convergence but the complete Pareto-optimal frontier is not discovered. For functions, UF5 and UF6, RSMOGA fails to converge to Pareto-optimal front. Both the functions have discontinuous Pareto front.

The objective function profile is multi-modal near the global Pareto-optimal frontier, and a slight perturbation in the optimization variables causes the solutions to become dominated. Also, the phenomenon of genetic drift causes the population to follow the good solutions, which get discovered early in the search process. This genetic drift results in the clustering of the solutions around these points. Further Rank-sum technique fails on three objective functions UF8 and UF9. The reason is multiple solution share same rank as number of objective and size of population increases, the discrimination among the solution becomes difficult. Even the use of explicit diversity maintenance technique won’t serve the purpose and hence the algorithm fails to converge on the true Pareto-optimal front. RSMOGA has shown good performance on few bi-objective functions having convex, non-convex and continuous Pareto fronts.

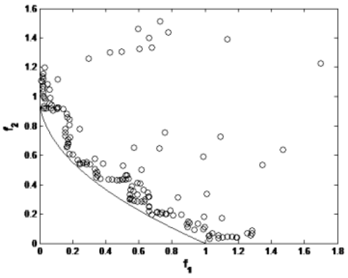

Fig.3. 2 Best approximate to UF1 (RSMOGA)

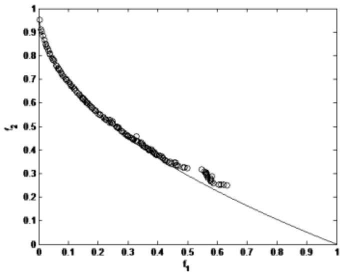

Fig. 3.3 Best approximate to UF2 (RSMOGA)

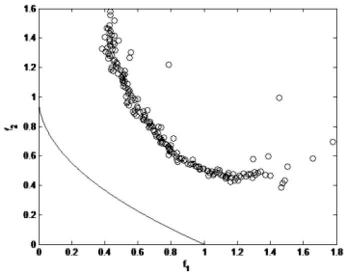

Fig. 3.4 Best approximate to UF3 (RSMOGA)

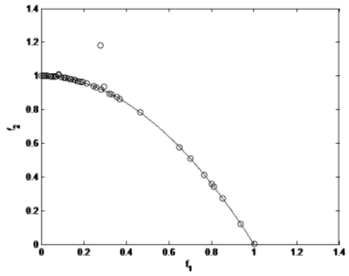

Fig.3. 5 Best approximate to UF4 (RSMOGA)

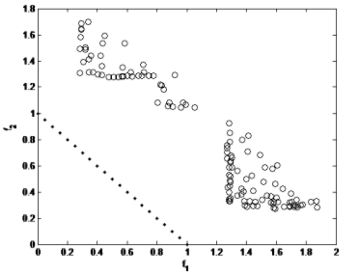

Fig 3.6 Best approximate to UF5 (RSMOGA)

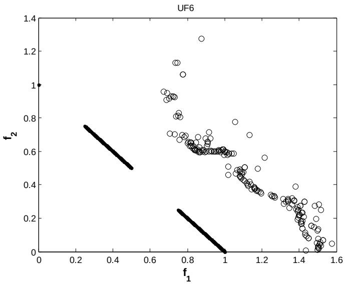

Fig 3.7 Best approximate to UF6 (RSMOGA)

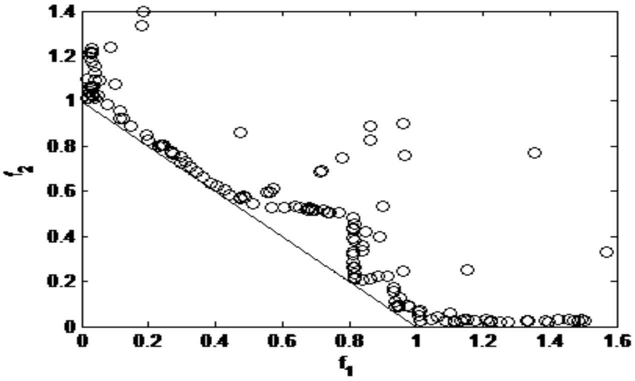

Fig. 3.8 Best approximate to UF7 (RSMOGA)

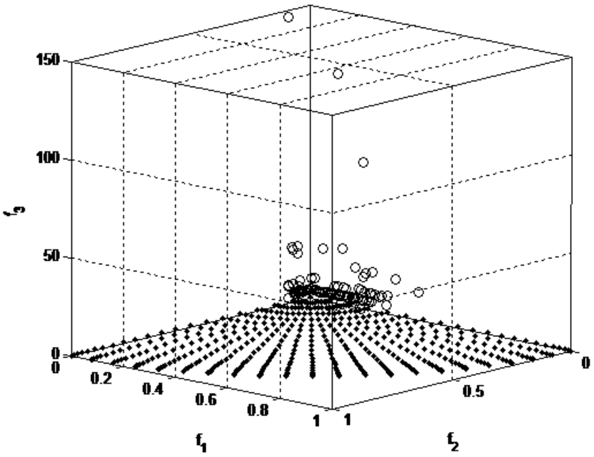

Fig. 3.9 Best approximate to UF8 (RSMOGA)

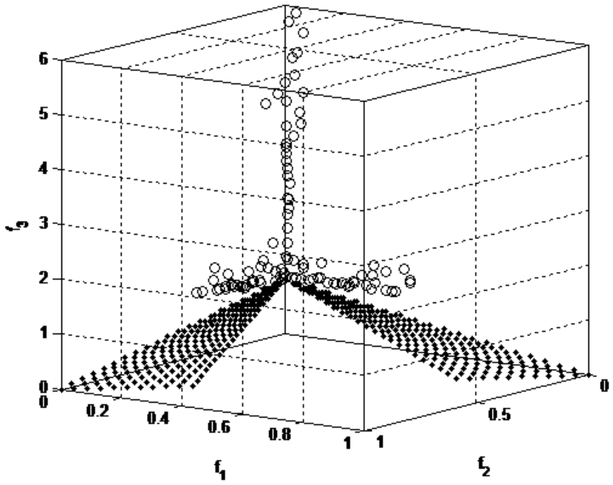

Fig. 3.10. Best approximate to UF9 (RSMOGA)

Part B: 30 random simulations are performed for each problem with SNOVMOGA. The minimum (Min), maximum (Max), mean, and standard deviation (Std) of the IGD metric are reported in Table 3.5 and figure 3.11-3.19 shows plot of approximate solutions for function UF1-UF9 with SNOVMOGA. For seven bi-objective instances, SNOVMOGA found good approximation to UF2, UF3, UF4 and UF7 but performed poorly on UF5 and UF6. It has given comparable results for UF1.For three objective instances; SNOVMOGA has given comparable performance for UF8 and shown good performance for UF9. The algorithm fails on functions having discontinuous Pareto fronts.

The possibility of different rank (SNOV) assigned to every solution in the population is very high. So discrimination among the solution at the time of selection as parents becomes easy and hence a good (fit) and divorced set of parents leads to improved performance of the algorithm in terms of convergence and diversity. The proof of the same is given in table 3.5 in the form of IGD values. Discrimination among the solution is not difficult even the number of objective increases.

TABLE 3.5 THE IGD METRIC FOR SNOVMOGA

Problem	Min	Max	Mean	Std
UF1	0.040123	1.000742	0.0573916	0.085339
UF2	0.009862	0.526735	0.0110476	0.016462
UF3	0.022385	0.044631	0.0311205	0.004444
UF4	0.012623	0.059979	0.0162622	0.006118
UF5	0.774876	1.776223	0.8196147	0.407084
UF6	0.265128	1.838803	0.3116606	0.289605
UF7	0.009107	0.669267	0.0104281	0.013800
UF8	0.225452	1.956260	0.3225753	0.323283
UF9	0.250592	1.00486	0.1084608	0.1806802

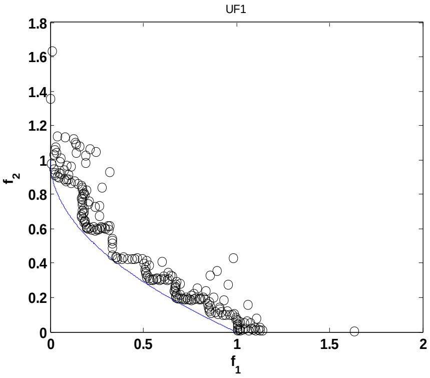

Fig. 3.11. Best approximate to UF1 (SNOVMOGA)

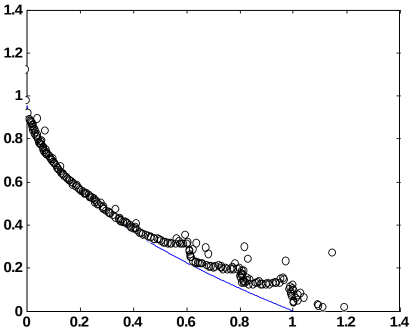

Fig. 3.12 Best approximate to UF2 (SNOVMOGA)

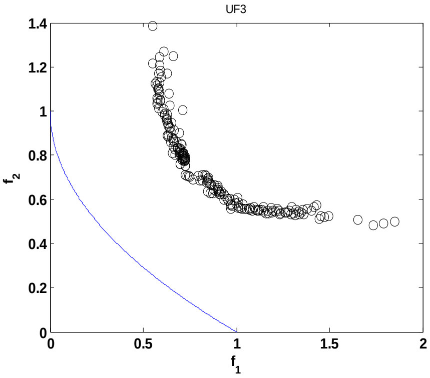

Fig. 3.13. Best approximate to UF3 (SNOVMOGA) (SNOVMOGA)

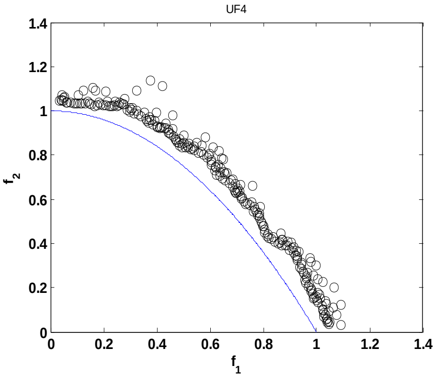

Fig. 3.14. Best approximate to UF4 (SNOVMOGA)

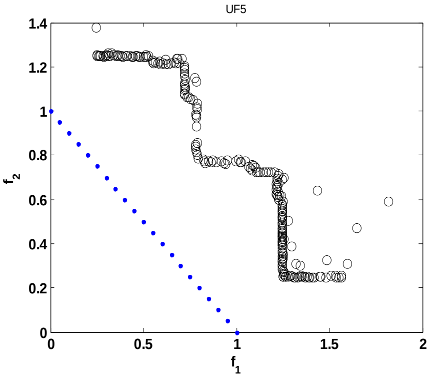

Fig.3.15.Bestapproximateto UF5 (SNOVMOGA)

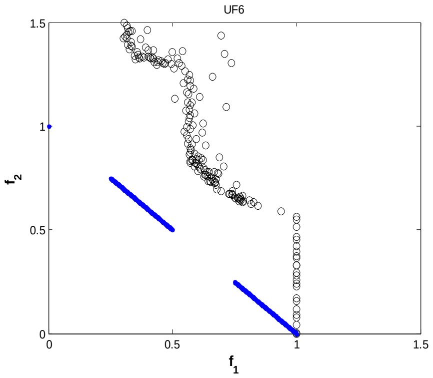

Fig. 3.16. Best approximate to UF6 (SNOVMOGA)

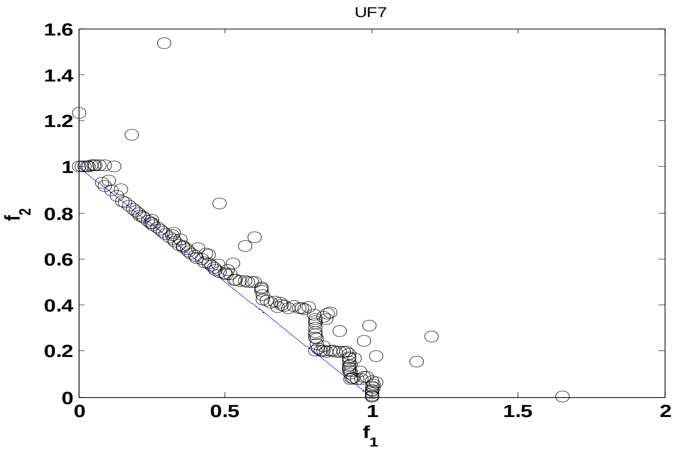

Fig. 3.17. Best approximate to UF7 (SNOVMOGA)

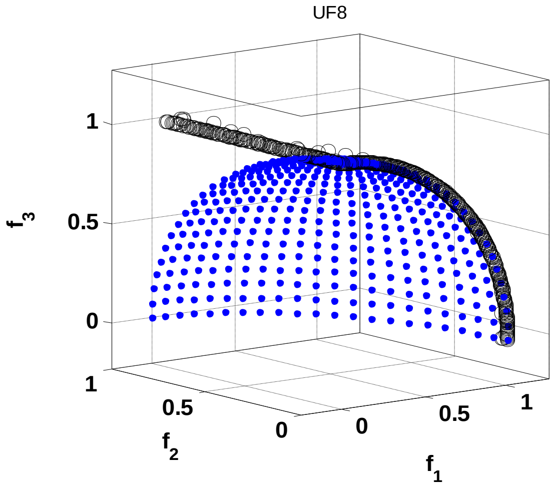

Fig. 3.18. Best approximate to UF8 (SNOVMOGA)

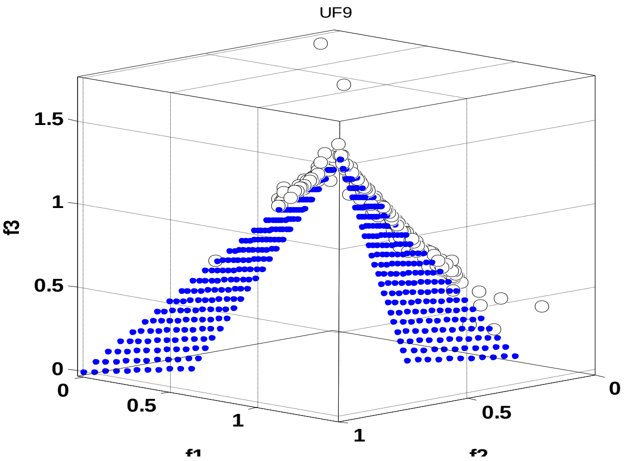

Fig. 3.19. Best approximate to UF9 (SNOVMOGA)

TABLE 3.6 COMPARISON OF CPU TIME OF RANK-SUM, SNOV AND NONDOMINATED SORTING TECHNIQUES

Problem	**Rank-sum Sort (in seconds)**	**SNOV Sort (in seconds)**	**Non-dominated Sort (in seconds)**
UF1	**61.984**	341.922	1008.341
UF2	**32.754**	377.031	1014.821
UF3	**53.016**	362.673	1028.630
UF4	**34.890**	310.475	1002.473
UF5	**168.892**	390.539	1068.459
UF6	**54.782**	392.531	1015.378
UF7	**53.407**	320.569	1013.489
UF8	**60.062**	358.638	1104.672
UF9	**76.625**	360.521	1100.592

3.5 Performance Comparison

3.5.1 Simulation Speed

Further experiments are performed to find computational efficiency of the proposed techniques. For the Rank-sum and SNOV method, it requires O(N) comparisons to find the maximum and minimum values and O(N) *comparisons* to identify the corresponding bins for each objectives. In total, the complexity of each methods is O(MN), where M is the number of objectives and N is the number of solutions. From the complexity calculations, we can observe that the complexities of the Rank-sum and SNOV methods are in linear form while non-domination sorting is non-linear. This will reduce the CPU time of the RSMOGA and SNOVMOGA. For the commonly used non-domination sorting, the complexity to obtain the overall non-dominated set is O (MN^2).

Table 3.6 shows the CPU time (in seconds) taken by the Rank-sum, SNOV and nondominated sorting techniques on problems UF1-UF9. From the table data it is clear that Rank-sum sorting comes out to be winner among the three techniques. Rank-sum sorting is a simple technique, which makes use of simple aggregation function to assign scalar fitness to the individuals in the population and a less complex diversified selection scheme for formation of next generation of population. Less number of scans for fitness assignment and selection of individuals has greatly helped in reduction in computation time and hence enhancement in the efficiency of the algorithm.

SNOV sorting has outperformed nondominated sort but it is less efficient than Rank-sum sort. Reason for it is that though SNOV use simple aggregation function to find sum of normalized objective value and assign the sum as scalar fitness to the individuals so that they can be discriminated at time of selection for formation of next generation population. Further SNOV used improved diversified selection, which involves a pre-selection method. This pre-selection method is an overhead as far as efficiency is concern otherwise has a great advantage in selection of trade-off solutions. Number of iteration increases in pre-selection method and hence SNOVMOGA requires more CPU time than RSMOGA but less CPU time than non-dominated sort. It is proved that the proposed non-Pareto based techniques are computationally efficient than Pareto-dominance based techniques.

3.5.2 Rank distribution

Corne and Knowles [69] proposed to measure the relative entropy of the distribution of ranks induced by a method in order to analyze its effectiveness. They described this measure as follows: consider a population of N ranked solutions (there are at most N ranks and at least 1). The relative entropy of the distribution of ranks D is give by:

$$re(D) = \frac{\sum_{r} \frac{D(r)}{N} \log\left(\frac{D(r)}{N}\right)}{\log\left(1/N\right)}$$

Where D(r) denotes the number of solutions with rank r. re(D) tends to 1 as we approach to the ideal situation in which each solution has a different rank. On the other hand, when all individuals share the same ranking position, re(D) is equal to zero. Thus, it is supposed that a ranking method providing a richer ordering would lead to a better performing optimization scheme. For experimentation population size of 1000 is taken and function used is UF1. Figure 3.21 shows the relative entropy of the ranking distribution produced by the studied methods. Results shown in the figure are average performance of 30 runs.

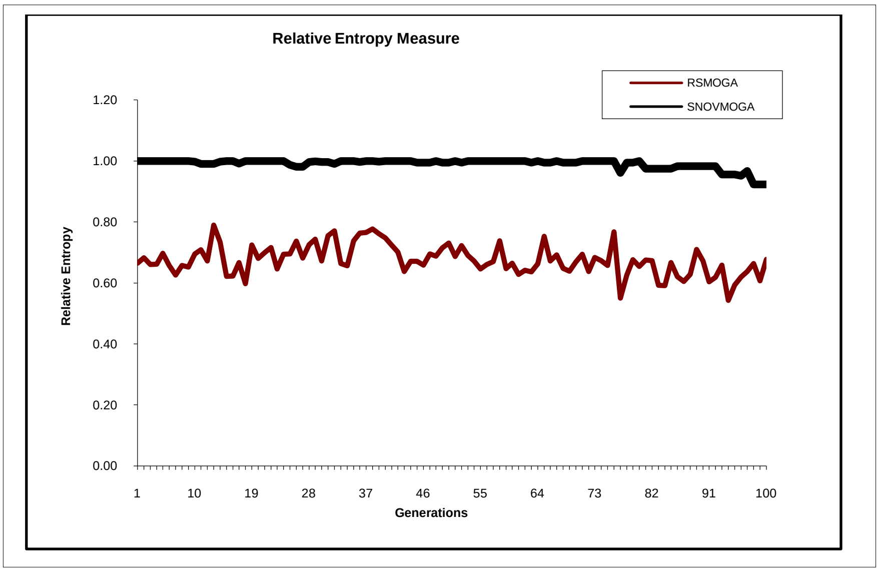

Fig.3. 20. Comparison of Relative Entropy of RSMOGA with SNOVMOGA

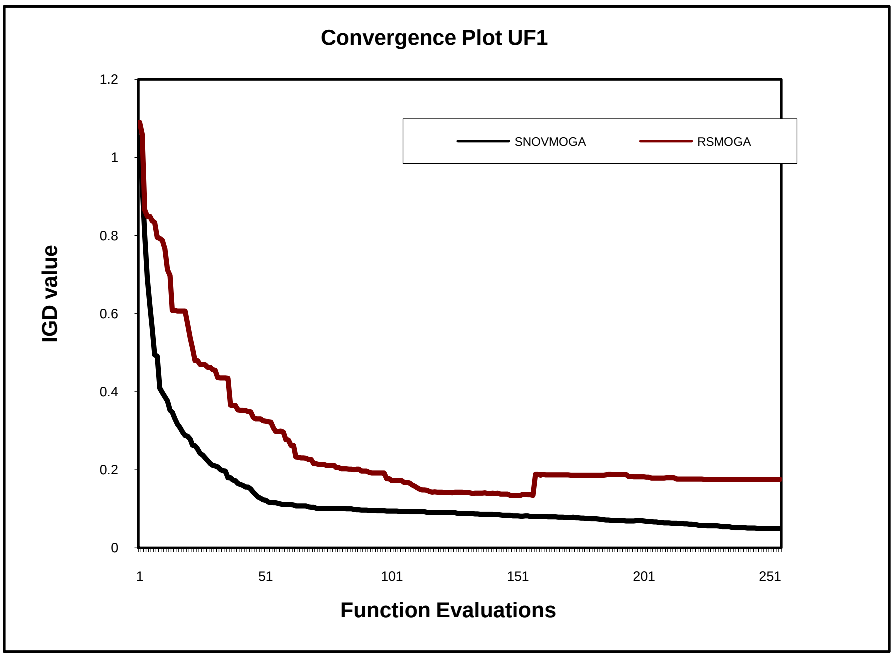

Fig. 3.21 Comparison of convergence and diversity of RSMOGA and SNOVMOGA for UF1 function

From figure 3.21 it is clear that SNOV has better rank distribution than Rank-sum technique. The reason behind this performance is that SNOV technique assigns unique SNOV (Rank) to every solution in the population whereas Rank-sum assigns same rank to multiple solutions and hence the ranking is less ordered. SNOV technique is fine-grained ranking technique.

3.5.3 Convergence and diversity

Experiments are performed to show that how effectively the two proposed schemes handle the issue of convergence and diversity. The figure 3.22 shows plot of IGD metric for UF1 function for RSMOGA and SNOVMOGA. It is clear from the figure that SNOV based ranking and diversity maintenance scheme have given better convergence and diversity than the Rank-sum based scheme. The reason is the fine-grain ranking of SNOV and the reference point based selection (pre-selection) of individuals for formation of next generation population. For all the functions SNOVMOGA has given better convergence and diversity than RSMOGA.

3.5.4 Comparison with other algorithms

The hybrid AMGA (Archive-based Micro Genetic Algorithm) [73] is a constrained multi-objective evolutionary optimization algorithm. For the purpose of selection, AMGA uses a two-tier fitness assignment mechanism; the primary fitness is the rank, which is based on the domination level, and the secondary fitness is based on the diversity of the solutions in the entire population. The gradient-based local optimizer used with AMGA is the Sequential Quadratic Programming (SQP) algorithm. SQP is used to speed-up the search process and obtain an improvement in the objective function value as quickly as possible.

In [70], Multi-objective evolutionary programming (MOEP) using fuzzy Rank-sum with diversified selection is introduced. The performances of this algorithm as well as MOEP with non-domination sorting on the set of benchmark functions provided for CEC2009 Special Session and competition on Multi-objective Optimization are reported.

Performance of RSMOGA and SNOVMOGA is compared with hybrid AMGA and MOEP in Table 3.7. The table shows mean IGD of the four algorithms.

RSMOGA has outperformed the two algorithms AMGA and MOEP, on UF3, UF4 and UF7 whereas given similar results for UF1. For functions UF5 and UF6 RSMOGA fails, so the primary cause of this behavior is the objective function profile, which is multi-modal near the global Pareto-optimal frontier and a slight perturbation in the optimization variables causes the solutions to become dominated. Also, the phenomenon of genetic drift causes the population to follow the good solutions, which get discovered early in the search process. This genetic drift results in the clustering of the solutions around these points. From the obvious discussion we can comment that RSMOGA has performed well on functions having convex, non-convex and continuous Pareto fronts but fails on discontinuous Pareto fronts.

TABLE 3.7 COMPARISON OF SNOVMOGA WITH RSMOGA , AMGA AND MOEP

Problem	**SNOVMOGA**	**RSMOGA**	**AMGA**	**MOEP**
	MeanIGD	**MeanIGD**	**MeanIGD**	**MeanIGD**
UF1	0.057391	0.064189	**0.035886**	0.059604
UF2	**0.011047**	0.026720	0.016236	0.018911
UF3	**0.031120**	0.031159	0.069981	0.099172
UF4	**0.016262**	0.028890	0.040621	0.042761
UF5	0.819614	0.978930	**0.094057**	0.224524
UF6	0.311660	0.370197	0.129425	**0.103114**
UF7	**0.010428**	0.019931	0.057076	0.019733
UF8	**0.322575**	1.836091	**0.171251**	0.423023
UF9	**0.108460**	0.988287	0.188611	0.342012

SNOVMOGA has outperformed RSMOGA in almost all the test functions. SNOVMOGA has given good performance for function like UF1, UF2, UF4, UF7, UF8, and UF9 when compared with MOEP, whereas on functions UF2, UF3, UF4, UF7 and UF9, SNOVMOGA has shown improvement over hybrid AMGA.

3.6 Key Findings

In this chapter, two non-Pareto based selection approaches, Rank-sum and Summation of Normalized Objective Value as an alternative to Pareto-dominance based selection have been reviewed. Two multi-objective Genetic algorithm RSMOGA and SNOVMOGA based on above-mentioned selection approaches have been introduced, implemented and tested on unconstrained test problems of CEC09 test suit. Experiments have been performed to investigate the search capability of these algorithms and the effectiveness of the two approaches in handling the issues of convergence and diversity in multi-objective optimization problems. Also the two methods are compared with Pareto-dominance based non-dominated sorting technique.

Experimental results presented in the chapter indicate that the proposed algorithms are able to guide the search process towards the optimum for the seven bi-objective and the two 3-objective test functions. SNOV based selection technique is fine-grained ranking technique and provides high order ranking and has helped the SNOVMOGA to give better convergence and diversity in comparison to RSMOGA and non-dominated sort. On the other hand RSMOGA has shown better computational efficiency than SNOVMOGA and non-dominated sort. Rank-sum sort is less complex technique and adaptive diversified selection requires less iteration for selection of solutions for next generation.

Both RSMOGA and SNOVMOGA have shown better convergence and diversity in all test functions having continuous convex and nonconvex Pareto fronts but performed poorly on functions having discontinuous Pareto fronts.

Chapter 4

Decomposition based MOGA with Opposition Based Learning (OBL)

MOP can be solved either by considering MOP as a whole or by using decomposition methods which solves scalar optimization sub problems simultaneously by evolving a population of solutions. There are several approaches for converting the problem of approximation of the PF into a number of scalar optimization problems. This chapter describes two decomposition approaches namely weighted sum approach and Tchebycheffs scalarization approach. These approaches require uniformly distributed weight vectors to maintain diversity among solutions. In this chapter we introduce decomposition based multi-objective genetic algorithm with Opposition operation. In this work Opposition Based Learning (OBL) concept is used in a unique way for uniform weight vector generation. Also to have diversity among solutions and proper exploration of search space opposition based learning concept is used for population initialization and both parent and opposite parent are allowed to reproduce. Chapter covers topics like experimental details, results and performance comparison with other algorithms. Salient findings of the study are at the end of the chapter.

4.1 Decomposition based Approach

It is well-known that a Pareto optimal solution to a MOP, under mild conditions, could be an optimal solution of a scalar optimization problem in which the objective is an aggregation of all the objective function values. Therefore, approximation of the PF can be decomposed into a number of scalar objective optimization sub problems. This is a basic idea behind many traditional mathematical programming methods for approximating the PF. Several methods for constructing aggregation functions can be found in the literature (e.g., [8]). The most popular ones among them include the weighted sum approach and Tchebycheffs approach.[1]

4.1.1 Tchebycheffs scalarization method

In Tchebycheff approach, the scalar optimization problem is in the form

$$\begin{aligned} &\textit{Minimize} \quad g^{te}\left(x \middle| \lambda, z^*\right) = \max_{1 \le i \le m}\left\{\lambda_i \left| f_i(x) - z_i^* \right|\right\} \\ &\textit{Subject to} \quad .x \in \Omega \end{aligned}$$

Where $z^*=(z_1^*,\ldots,z_m^*)^T$ is the reference point, i.e., $z_i^* = \max\{f_i(x)|x \varepsilon \Omega\}$ for each $i = 1,\ldots,m$.

For each Pareto optimal point x^*there exists a weight vector λ such that is the optimal solution of Tchebycheffs function and each optimal solution of scalar optimization problem is a Pareto optimal solution of MOP. Therefore, one is able to obtain different Pareto optimal solutions by altering the weight vector. One weakness with this approach is that its aggregation function is not smooth for a continuous MOP. This scalarization approach is use in the proposed algorithm.

4.1.2 Weight-sum scalarization method

This approach considers a convex combination of the different objectives. Let $\lambda = (\lambda_1,\ldots,\lambda_m)$ (be a weight vector, i.e., $\lambda_i \geq 0$ for all i = 1,…,m and $\sum_{i=1}^{m} \lambda_i = 1$. Then, the optimal solution to the following scalar optimization problem: is a Pareto optimal point to (1), where we use to emphasize that is a coefficient vector in this objective function, while is the variables to be optimized. To generate a set of different Pareto optimal vectors, one can use different weight vectors in the above scalar optimization problem. If the PF is concave (convex in the case of minimization), this approach could work well. However, not every Pareto optimal vector can be obtained by this approach in the case of nonconcave PFs.

$$\begin{aligned} &\max\textit{imize} \ \ g^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \\ &\textit{subject to} \ \ x \in \Omega \end{aligned}$$

To overcome these shortcomings, some effort has been made to incorporate other techniques such as -constraint into this approach, more details can be found in [1].

4.2 Opposition Based Learning (OBL)

Tizhoosh [74] introduced the concept of Opposition-Based Learning (OBL). The basic idea behind OBL is that whenever we seek the solution in a direction that is beneficial to consider the opposite direction as well. Many machine intelligence algorithms consider finding the solution of a given problem as function approximation. Thus, if the objective is to search for the solution x, the algorithm makes an estimation ^x which should resemble the closest value to x. Such algorithms can be computationally expensive if the required solution must be very accurate. Starting point of search can dramatically affect the accuracy of the found solution (among others due to local maxima or minima) and the convergence time. In many cases, the starting points are chosen randomly, such as weights of a neural network, initial population of evolutionary algorithms, and action policy of reinforcement agents. If the starting point is close to the optimal solution, this results a faster convergence. On the other hand, if it is very far from the optimal solution, such as opposite location in worst case, the convergence will take much more time or even the solution can be intractable. Looking simultaneously for a better candidate solution in both current and opposite directions may help to solve the problem efficiently [75]. The opposite operation used in differential evolution for solving single-objective problems has been demonstrated effectively in paper [76]. In the following, we give the definition of the opposite number.

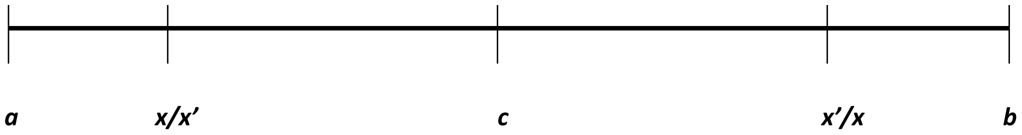

Fig. 4.1 shows number and its opposite number in 1-D

Definition (Opposite Number): Let $x \in [a, b]$ be a real number. The opposite number $\sim x$ is defined by $\sim x = a + b - x$.

Similarly, this definition can be extended to higher dimensions as follows.

Definition (Opposite Point): Let $P = (x_1, x_2, \cdot\cdot\cdot, x_n)$ be a point in n-dimension space, where $x_1, x_2, \cdot\cdot\cdot, x_n \in R$ and $xi \in [a_i, b_i], i = 1, 2, \cdot\cdot\cdot, n$.

$$\overline{X}_i = a_i + b_i - X_i$$

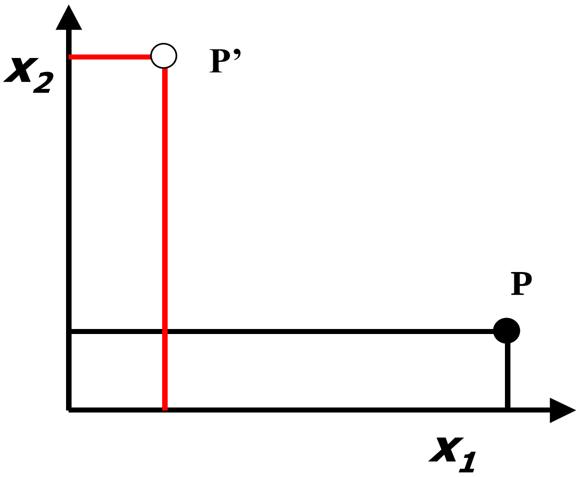

Fig 4.2 shows number and opposite number in 2-D space

The opposite point $\sim P = (\sim x_1, \sim x_2, \cdot\cdot\cdot, \sim x_n)$ is defined by its components Let $P(x_1,x_2,\ldots,x_n)$ be a point in n-dimensional space with x_i Є $[a_i, b_i]$; i Є$\{1,2,\ldots,n\}$ be a candidate solution. Assume f(x) is a fitness function which is used to measure candidate optimality. According to opposite point definition, $P'(x_1',x_2',\ldots,x_n')$ is the opposite of $P(x_1,x_2,\ldots,x_n)$. Hence the point and its opposite point are evaluated simultaneously to continue with the fitter one.

The definition of OBL is making two fundamental assumptions:

1) One of the estimate or the opposite-estimate is always closer to the solution (but fitness can mislead!),
2) Considering the opposition is more beneficial than generating independent random solution and taking the best among them.

Opposition based learning used in Evolutionary algorithm: In 2006, Rahnamayan et al. [76] applied the opposition concept to initialize Evolutionary Algorithms. In this method, an initial uniform-random population is generated. Then the opposite population is calculated and from the union of the two populations, the fittest candidate-solutions are selected. This supports more diversity and exploration when starting the search process. Rahnamayan et al. (2007) introduced Quasi-Oppositional population initialization method in which, the opposite population consists of uniform-random points generated in the interval/region between the center (middle) of the search space and the opposite point [77]. Rahnamayan and Wang [2008] introduced Center-Point and Center-Based sampling methods to enhance population-based algorithms [78]. Peng and Wang in 2010 introduced the Uniform-Quasi-Opposite Different Evolution (UQODE) algorithm [79]. Rahnamayan et al base this algorithm on Quasi-Opposite Differential Evolution (QODE)[80]. In the UQODE algorithm, the uniform design is used for generating the first population. Malisia and Tizhoosh [82] apply the concept of opposition to Ant Colony Optimization (ACO). Han and He introduced the algorithm of Opposition-based PSO (OPSO) for solving noisy problems [81].

4.3 Weight Vector generation using OBL

It is found that MOEA/D might not work very well if the solutions to neighboring sub problems are not very close in the decision space. Appropriate setting of weight vectors lead to well separated solution set. However, it is often hard, if it is not impossible, to know beforehand which setting is proper. A possible solution may be to tune weight vectors adaptively based on the information collected during the search.

Very often since the objectives in MOP contradict each other no point in search space minimizes all the objectives simultaneously. One has to balance them. The best tradeoffs among the objectives can be defined in terms of Pareto optimality. Any improvement in a Pareto optimal point in one objective must lead to deterioration in at least one other objective. With the above idea we suggest opposition operation can be used for uniform weight vector generation. Objective values of Pareto optimal points are opposite in nature i. e. (0,1),(0.3,0.7),……,(1,0) and sum of objective weights of a solution equals one. In this work we have used opposition based learning concept for weight vector generation. The method is as follows:

```
Uniformly linearly divide the range [0, 1] into N values.
Store these N values as first component of N weight vectors.
For i=1 to N
    Calculate opposite of $i^{th}$ value i.e.first component of $i^{th}$ weight vector using opposite operation and assign it as second component of $i^{th}$ weight vector.
End
```

For example Number of weight vectors N = 11, Lower bound a = 0, Upper bound b =1, We linearly divide range [0,1] into 11 values. Therefore N={0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1}

Assign these values to First component of N weight vectors. Using the opposition operation we can calculate second component of weight vector as opposite number of first component. If first component of second weight vector is 0.1 then Second component = 0+1-0.1=0.9. Thus $\lambda_1(0,1)$, $\lambda_2(0.1,0.9)$,…, $\lambda_{11}(1,0)$.

This technique of weight vector generation is unique and simple. With this method we can generate uniform weight vectors for two objectives MOP in an efficient manner. Weight vector generation given in MOEA/D is very complicated since large number of random weights is generated and then one has to find distance between these weight vectors which is computationally complex method. In some papers (eg. [87]) weight vectors are randomly generated. Since weight vectors acts like pivot and well spread weight vectors help in finding well spread approximate solutions, opposition based weight vector generation will help MOGA to get good approximate to Pareto front.

4.4 DMOGA-OBL

In this section we introduce decomposition based MOGA with opposition operation. A Pareto optimal solution to an MOP could be an optimal solution of a single objective optimization problem in which the objective is a linear or nonlinear aggregation function of all the individual objectives. Therefore, approximation of the PF can be decomposed into a number of single objective optimization problems. Some MOEAs such as MOGLS [83]–[85] and MSOPS [86] adopt this idea to some extent. MOEA/D [19] (Multi-Objective Evolutionary Algorithm based on Decomposition) is a very recent evolutionary algorithm for multi-objective optimization using the decomposition idea. MOEA/D explicitly decomposes the MOP into N scalar optimization sub problems. It solves these sub problems simultaneously by evolving a population of solutions. At each generation, the population is composed of the best solution found so far (i.e. since the start of the run of the algorithm) for each sub problem. The neighborhood relations among these sub problems are defined based on the distances between their aggregation coefficient vectors. The optimal solutions to two neighboring sub problems should be very similar. Each sub problem (i.e., scalar aggregation function) is optimized in MOEA/D by using information, only from its neighboring sub problems. Paper [87] presents MOTGA (Multiple objective Tchebycheffs based Genetic Algorithm) a new multi-objective genetic algorithm based on the Tchebycheffs scalarising function, which aims to generate a good approximation of the nondominated solution set of the multi-objective problem. The algorithm performs several stages, each one intended for searching potentially nondominated solutions in a different part of the Pareto front. Pre-defined weight vectors act as pivots to define the weighted-Tchebycheffs scalarising functions used in each stage. These decomposition based MOEAs use random weight vectors.

In this work opposition based learning is used for weight vector generation which generates uniformly distributed weight vectors. Also this algorithm makes use of opposition operation for population initialization and both parent and opposite parent are given chance to reproduce so that more diverse solution will be produced. Tchebycheffs scalarization (TS) function is used for scalar fitness assignment to a solution. A solution having minimum TS value with respect to weight vector λ and reference point is selected for next generation. For reproduction SBX crossover operator is used. First Parent solution is sequentially selected from parent population and second parent is randomly selected from opposite parent population. Polynomial mutation operator is used for mutation.

Pseudo code of proposed DMOGA-OBL is as follows:

```
1. Initialize the population $P$ with $N$ individuals randomly in the
   decision space [$a, b$]; & evaluate
2. Initialize reference point with minimum objective values
3. Initialize Final Set, FS = $\phi$
4. Generate N weight vectors using opposition based learning concept.
5. For gen =1 to Maxgen
     Calculate the opposite population O$P$ by opposite operation
     Apply SBX operator on population P and opposite population OP and generate
         offspring population OFP of size 2N
     Combine P, OP and OFP (of size 4N)
     Update reference point with minimum objective values
     For i=1 to N
         o  Select randomly (0.1*(4N)) solutions from combined population.
         o  Calculate scalar fitness value of each selected solution using Tchebycheff
            scalarising (TS) function for $i^{th}$ weight vector.
         o  Add solution having minimum TS value to FS
     End
     Copy solutions in FS to P
   End
6. Present all the solution in FS as best approximate solution and stop.
```

4.5 Experimental design and results

For experimentation we have implemented the GA procedure using MatLab. The test problems (UF1-UF7) in this work are taken from CEC2009 special session and competition.

Parameter setting for experimentation:

Population size (N): 100 (for 2 objectives)

Number of generations: 1000

Maximum No of Function Evaluations: 100000

Crossover probability parameter (pc): 0.9

Mutation probability: 0.1

Probability Distribution indices for SBX : 20

Probability Distribution indices for mutation: 20

Number of Parents: 2

Number of children (λ): 2

30 random simulations are performed for each problem. The minimum (Min), maximum (Max), mean, and standard deviation (Std) of the IGD metric are reported in Table 4.1 and figure 4.3-4.10 shows plot of approximate solutions for function UF1-UF9.

TABLE 4.1 THE IGD METRIC

Problem	Min	Max	Mean	Std
UF1	0.003287	0.5007241	0.0099164	0.0089393
UF2	0.004362	0.0267322	0.0067606	0.0018622
UF3	0.022385	0.0446371	0.0311205	0.0404461
UF4	0.012623	0.0597914	0.0162621	0.0051184
UF5	0.094876	0.9762371	0.2906141	0.0607084
UF6	0.015128	0.9838392	0.0311686	0.0289605
UF7	0.004607	0.0696725	0.0048101	0.0013801
UF8	0.982545	4.956260	0.836727	0.145532
UF9	0.550592	5.966486	0.891271	0.134896

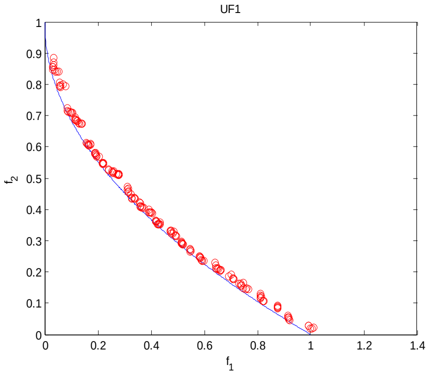

Fig. 4.3 Approximate set of UF1 function

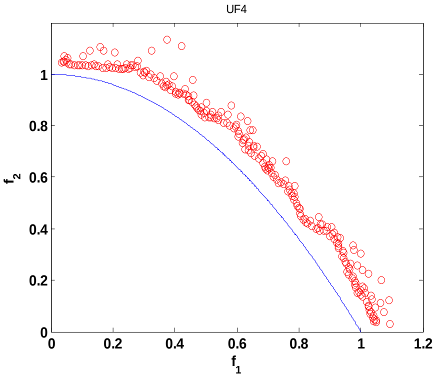

Fig. 4.6 Approximate set of UF4 function

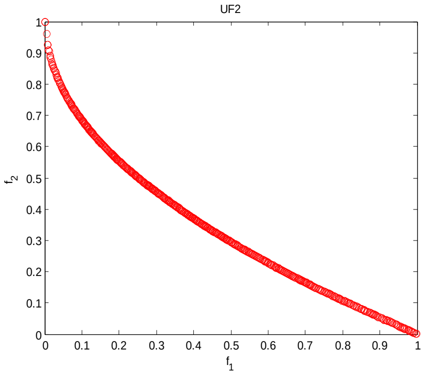

Fig. 4.4 Approximate set of UF2 function

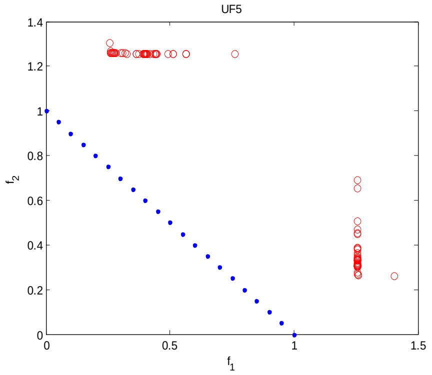

Fig. 4.7 Approximate set of UF5 function

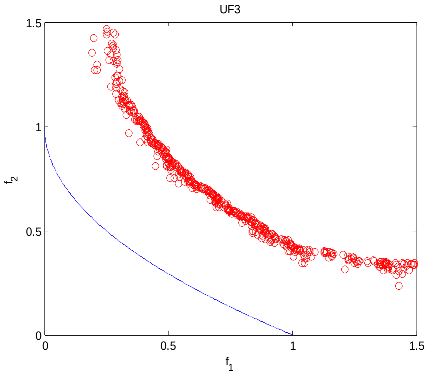

Fig. 4.5 Approximate set of UF3 function

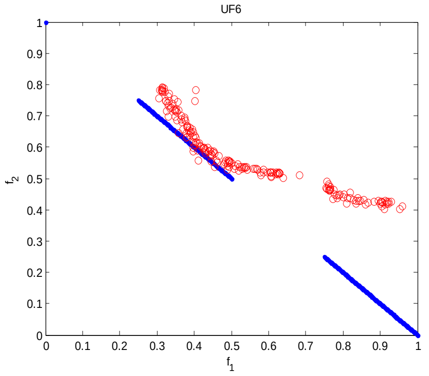

Fig. 4.8 Approximate set of UF6 function

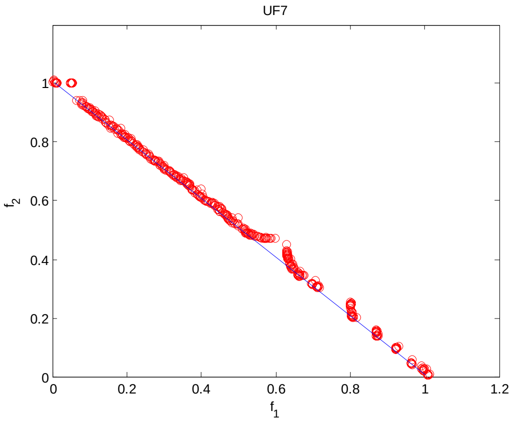

Fig. 4.9 Approximate set of UF7 function

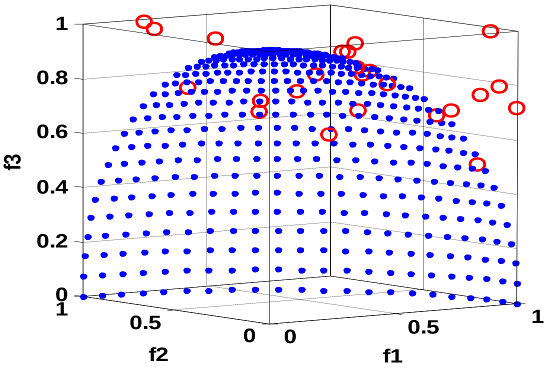

Fig. 4.10 Approximate set of UF8 function

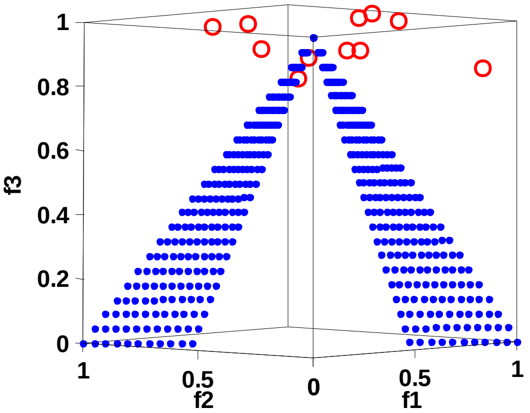

Fig. 4.11 Approximate set of UF9 function

4.6 Performance comparison with other algorithms

Performance of RSMOGA, SNOVMOGA and MOEA/D is compared with DMOGA-OBL in Table 4.2. The table shows mean IGD of the four algorithms. DMOGA-OBL when compared with RSMOGA and SNOVMOGA has performed better on UF1-UF7 but not shown improvement in UF8 & UF9. It has good improvement on two functions (UF5 & UF6) having discontinuous Pareto front. When compared with MOEA/D shown comparable performance in UF1, UF2, UF4 & UF7 but performed poorly on UF3, UF5, UF6, UF8 & UF9.

From the obvious discussion we can comment that DMOGA-OBL has performed well on functions having convex, non-convex and continuous Pareto fronts but fails on discontinuous Pareto fronts.

TABLE 4.2 COMPARISON OF DMOGA-OBL WITH SNOVMOGA, RSMOGA AND MOEA/D

Scheme	**RSMOGA**	**SNOVMOGA**	**DMOGA-OBL**	**MOEA/D**
Function	**MeanIGD**	***MeanIGD***	***MeanIGD***	***MeanIGD***
UF1	0.06418	0.05739	**0.00991**	**0.00435**
UF2	0.02672	0.01104	**0.00676**	0.00679
UF3	0.03115	0.03112	0.03127	**0.00742**
UF4	0.02889	0.01626	**0.01626**	0.06385
UF5	0.97893	0.81961	0.29061	**0.18071**
UF6	0.37019	0.31166	0.03116	**0.00587**
UF7	0.01993	0.01042	0.00481	**0.00444**
UF8	1.83609	0.32257	0.83672	**0.05840**
UF9	0.98828	0.10846	0.89127	**0.07896**

4.7 Key findings

This study has given insight into decomposition based approach of multi-objective optimization. A brief description of Tchebycheffs scalarization and weight-sum approach is presented. A novel technique of uniform weight vector generation using opposition based learning is introduced in this chapter. Decomposition based Multi-objective Genetic Algorithm with opposition based learning is proposed. In the proposed algorithm, Opposition Based Learning (OBL) concept is used in a unique way for weight vector generation, which has produced uniform well distributed weight vectors and the performance of the algorithm is improved. Also to have diversity among solutions and proper exploration of search space opposition based learning concept is used for population initialization. Both parent and opposite parent, are allowed to reproduce. The simulation results show the effectiveness of the proposed algorithm.

Chapter 5

Dying Strategy based MOGA

Genetic Algorithm (GA) mimics natural evolutionary process. Since dying of an organism is important part of natural evolutionary process, GA should have some mechanism for dying of solutions just like GA have crossover operator for birth of solutions. Also, in nature, occurrence of event of dying of an organism has some reasons like aging, disease, malnutrition and so on. In this chapter we introduce a novel thought of gradual dying of solution and making dying strategy as explicit part of evolutionary process. We design three strategies of dying or removal of solution from next generation population. In a new MOGA framework dying strategy is deployed. MOGA takes decision of removal of solution, based on one of these three strategies. Experimental results show that gradual dying of solution can boost performance of genetic algorithm. Performance of three dying strategies is compared.

5.1 Introduction

In biology and ecology, dying is the end of an organism. Dying is the permanent cessation of all biological functions that sustain a living organism. Phenomena which commonly bring about death include biological aging, predation, malnutrition, disease, suicide, murder and accidents or trauma resulting in terminal injury [88]. Contemporary evolutionary theory sees death as an important part of the process of natural selection. It is considered that organisms less adapted to their environment are more likely to die having produced fewer offspring, thereby reducing their contribution to the gene pool. The gene pool of a species or a population is the variety of genetic information in its living members. A large gene pool (extensive genetic diversity) is associated with robust populations that can survive bouts of intense selection. Meanwhile, low genetic diversity reduces the range of adaption possible. Replacing native with alien genes narrows genetic diversity within the original population, thereby increasing the chance of extinction [89].

Algorithms based on strategies of evolution are called as evolutionary algorithms. The decision of selection of solution becomes complicated in presence of multiple conflicting objectives. Every multi-objective evolutionary algorithm has two goals; convergence and diversity, so they need two different mechanisms for fulfillment of these goals. Algorithms like NSGA-II use non-dominated sorting and crowding distance based selection strategies for convergence and diversity respectively [17]. Many such multi-objective evolutionary algorithms with explicit mechanism for convergence and diversity control are found in the literature [17, 35, 43, 47, 63].

Selection strategy uses Pareto-dominance based [17] selection or single scalar value based [90] selection to select a multi-objective solution. Selected solutions are passed to next generation and they become parents in that generation. Solutions which are not selected die out or discarded from population and never re-appear. Dying of solution is inherent part of evolutionary process and selection strategies given in literature performs dying or removal implicitly. Solutions having better fitness produce fitter offspring and selection strategies are likely to select both parent and their offspring. Offspring solutions use cross over operator to inherit the properties of Parent solution present in first generation. In subsequent generations offspring solutions carry forward the good properties of parent solutions. Problem with this selection strategy is that, after few generation whole population is dominated by presence of few solutions from initial population and their offspring i.e. trail of very few solutions from initial population reach to final generation and most of the solutions die out somewhere in between generations.

In analogy with nature, if GA introduces a mechanism to control the rate of dying of solutions then diversity in population can be maintained. If dying rate of solutions is low, number of solutions having variety of genetic material will contribute in the formation of next generation population by producing diverse offspring.

In this work dying of solution is made explicit part of evolutionary process and three strategies for dying of solution are proposed and implemented. One of the three proposed dying strategies is used to deterministically remove the solution from next generation population. Impact of dying rate of solutions on the performance of GA is studied. Idea of gradual dying is modeled and a new framework of MOGA is used to demonstrate the same.

5.2 Strategies for Dying of Solution

A thought; dying of parent solution; opposite of selection of parent solution, is materialized in this work. Proposed three strategies of dying (Parent Removal (ParRem)) are given below.

ParRem1: Remove solution having minimum distance from one or more solutions of next generation population. In this strategy before removing a solution from next generation population distance between all the solutions is checked. Since similarity means uniformity and dissimilarity means diversity, according to this strategy similar solution should die out and dissimilar solution should remain in population in order to have diversity in population. The solution having minimum distance from its neighbor solution will be removed from next generation population. Distance between solutions is calculated in objective space and distance measure used is Euclidean distance. Pseudo code for distance calculation is given below.

```
Initialize min_d = 9999,
 for  j = 1 : N-1
      for   i =  j+1 : N
           d = distance between i^th parent  and j^th parent
           if   (d < min_d)
                min_d = d;
                     idx =i;
          end if
     endfor
 endfor
```

where N is population size, *min_d* is minimum distance between two parent solutions. *idx* is index of parent having minimum distance. A parent having *idx* index will be removed from next generation population. This scheme looks similar to crowding distance assignment scheme of NSGA-II but it is not. In NSGA-II for crowding distance calculation all the solutions in the front are sorted first and then crowding distance is calculated among solution and its two neighboring solutions. Here in ParRem1 sorting is not required. Distance from every solution to all the solutions in the population is calculated and then the solution having minimum distance with any of the solution will be removed.

ParRem2: Remove solution having maximum SNOV (Summation of Normalized Objective Value)[90]. In this strategy to make effective range of all the objective functions equal objective values are normalized. After normalization effective range of all the objective function will be zero to one. Assign SNOV to each solution in next generation population. The SNOV will be treated as single scalar fitness of the solution. Now remove solution having highest SNOV value (for minimization of objective). Pseudo code for SNOV calculation is given below.

```
for m = 1 : M (number of objectives)
Find the maximum and minimum objective values of the m^th objective and calculate the range of the m^th objective.
Normalize the m^th objective values of all members using the equation:
    $f_m(x) = \frac{f_m(x) - f_{\min}}{f_{\max} - f_{\min}}$
    where $f_m$ is the normalized m^th objective value.
end for
for i = 1 : N
    Sum all normalized objective values of the member to obtain a single value.
endfor
```

ParRem3: Remove solution having poor fertility count. Frequency of reproduction is an important parameter in determining species survival: an organism that dies young but leaves numerous offspring displays, according to Darwinian criteria, much greater fitness than a long-lived organism leaving only one [89].

In this strategy, parameter fertility_count keep record of frequency of reproduction of a parent solution. Initially when population is initialized, zero fertility_count is assigned to all the solution in the population. If the offspring produced by the parent solution is better than the one who produced it then fertility_count of parent solution is incremented by one. Now fertility_count of parent solution is assigned to better offspring solution. In next generation population, dying of a solution is insured on the basis of fertility_count of the solution. A solution having minimum fertility_count will be removed from next generation population. Pseudo code is given below.

- Assign initial fertility_count = 0 to all the solution in the initial population
- If offspring solution wins the multi-level tournament then fertility_count of first parent solution is incremented by 1 and assigned as fertility_count to winner offspring solution
- Sort next generation population according to fertility_count
- Remove parent having minimum fertility_count

5.3 MOGA with Dying strategies

In this section a new MOGA framework is presented which uses simple mechanism for formation of next generation population. Uniform-random method is used for population initialization. Opposition based learning [74] is used to tune initial population. In the beginning each initial solution is given a unique number as *parent number*. For crossover operation first parent is sequentially selected and second parent is selected randomly from current population. Crossover operator generates two offspring that form family with first parent. Multi-level tournament operator is used to select one solution from the family. Parent number of first parent is assigned to offspring solution. Cycles of selection-crossover-multilevel tournament selection repeats with each solution in current population.

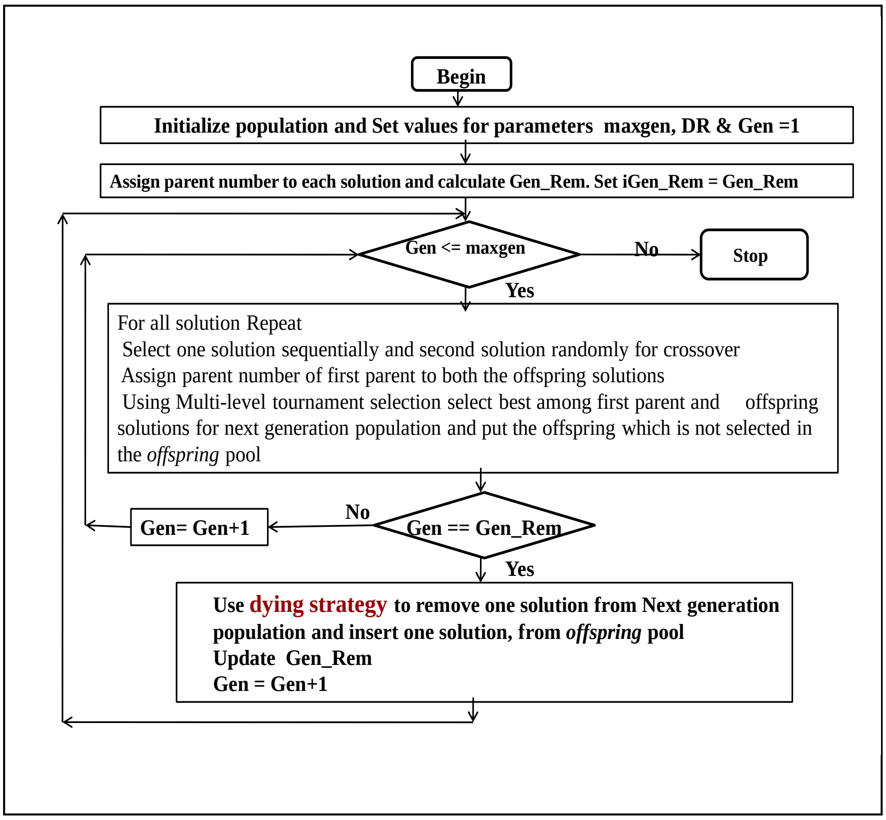
Begin
Initialize population and Set values for parameters maxgen, DR & Gen =1
Assign parent number to each solution and calculate Gen_Rem. Set iGen_Rem = Gen_Rem
Gen <= maxgen
No
Stop
Yes
For all solution Repeat
Select one solution sequentially and second solution randomly for crossover
Assign parent number of first parent to both the offspring solutions
Using Multi-level tournament selection select best among first parent and offspring solutions for next generation population and put the offspring which is not selected in the offspring pool
No
Gen= Gen+1
Gen == Gen_Rem
Yes
Use dying strategy to remove one solution from Next generation population and insert one solution, from offspring pool
Update Gen_Rem
Gen = Gen+1

Fig. 5.1. Proposed MOGA framework with dying strategy

Breaking analogy with nature, the algorithm deterministically decides when to remove a solution and which solution should die out. The decision of dying or removal of parent solution from next generation population is based on one of the three proposed dying strategies. Following formula decides the generation number in which a parent solution should be removed.

$$\text{Gen\_Rem} = \text{maxgen} / (N*DR)$$

Where maxgen is Maximum number of generation, DR is Dying Rate (in %) (Number of solutions to be removed from population) and N is size of population. For example If maxgen = 2000, DR = 30% and N = 200 then Gen_Rem = 2000 / (200*0.3) = 34. In 34th generation, one parent will be removed on the basis of one of the three strategies of dying, from next generation population. Now next generation contains 199 parent solutions. Dying of one solution makes space for new solution. This space is filled by introducing an offspring solution from offspring pool. Loser offspring of multilevel tournament selection forms the offspring pool. Next generation population contains 200 solutions but parents are 199 only. One solution is offspring of 199 selected solutions i.e. one solution have duplicate parent number. From 35th to 67th generations no solution will be removed and population evolves with 199 parents. In 34+34 =68th generation one more parent solution will be removed. In this way after every 34th generation one parent is removed from next generation population and in place of removed solution, one solution from offspring pool will be introduced in the next generation population in order to keep population size constant. At the end of maxgen generation final population contains traces of 140 solution and 60 (= 200*0.3) solutions are removed from population i.e. 70% parent solutions survive and 30% parent solutions die out. Algorithmic steps of proposed framework are given in fig 5.1.

Initialization and tuning the population: The most famous and widely-used initialization method is uniform-random initialization, also known as pseudo-random initialization. Even though pseudo-random initialization is computationally cheap, the quality of generated solutions is not very good and properly diverse.

In order to contest the uniform-random method, Tizhoosh introduced the opposition concept through Opposition-Based Learning. In this concept, the opposite of a guess or estimate (i.e., uniform-random) is calculated and compared to the original estimate (random). This has resulted in creation of successful initialization method for Metaheuristic algorithms. In 2006, Rahnamayan et al applied the opposition concept to initialize Evolutionary Algorithms. This supports more diversity and exploration when starting the search process [74].

In proposed algorithm opposition based learning is used to tune the population. A solution is sequentially selected from initial population and its opposite solution is calculated by using opposition based learning as given above. Now tournament is played between solution and opposite solution and winner is selected as tuned solution and loser is discarded. This procedure is repeated for every solution. If there are N solutions in the initial population then N tune solutions are generated.

Tournament selection: In this algorithm multi-level tournament selection is used. In first level of tournament selection, tournament is played between two offspring solutions. In second level, tournament is played between winner offspring solution and First parent. The winner of second level of tournament is selected as best solution and becomes member of next generation population.

5.4 Experimental design and results

The proposed framework for MOGA with three strategies for dying of solution is coded in MatLab 7.1. 30 independent runs have been taken for each problem with different strategies. Experimental parameter settings used are:

Population size (N) = 100
Maximum no. of generation (maxgen) = 3000
Dying Rate (DR) = 10%, 30%, 40%, 50%, 60%, 90%
Probability Distribution index for MPX crossover operator = 1
Number of Parents= 2
Number of children = 2

In this section discussion is on experimental results of MOGA with three strategies of dying of solution.

Strategy ParRem1: The Performance of dying strategy ParRem1 in terms of MeanIGD & SPREAD on function UF1-UF9 is shown in Table 5.1. Among nine functions (UF1-UF9) ParRem1 has given lowest value of SPREAD Metric (**0.40815) for UF2 for dying rate = 50%.** Fig 5.2 shows plot of SPREAD produced by ParRem1 for functions UF1-UF9 with different Dying rates. It is observed from the figure that a moderate dying rate (50%) has given low value for SPREAD metric for all functions. A low value of SPREAD means better diversity among solutions of nondominated set. For functions UF1, UF2 & UF8 best MeanIGD value is obtained for 50% dying rate. For rest of the functions best MeanIGD is for low dying rate.

Strategy ParRem2: The Performance of dying strategy ParRem2 in terms of MeanIGD & SPREAD on function UF1-UF9 is shown in Table 5.2. For functions UF2 & UF8 best MeanIGD value is obtained at 50% dying rate. For rest of the functions best MeanIGD is for low dying rate.

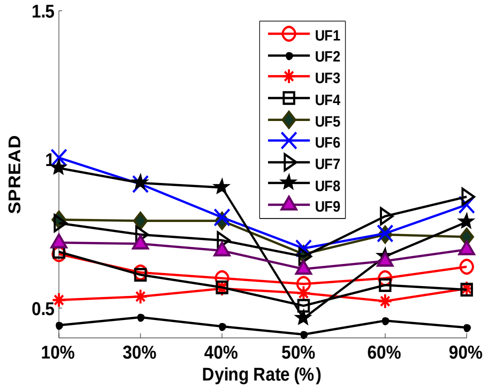

Fig 5.2 shows plot of SPREAD produced by ParRem1 for UF1-UF9

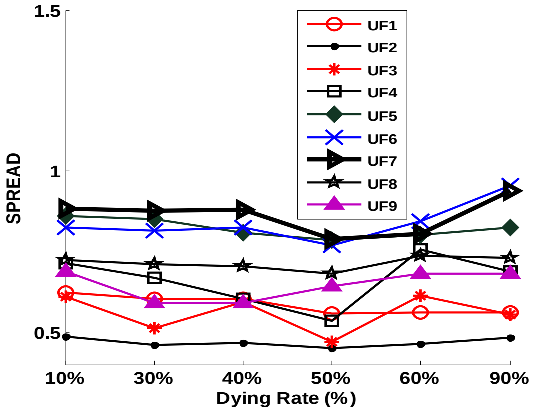

Fig 5.3 shows plot of SPREAD produced by ParRem2 for UF1-UF9

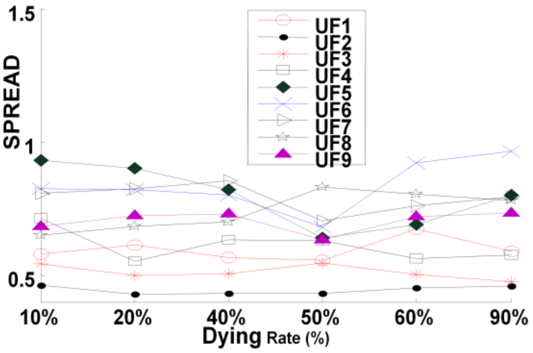

Fig 5.4 shows plot of SPREAD produced by ParRem3 for UF1-UF9

TABLE 5.1 PERFORMANCE OF PARREM1ON UF1-UF9 (POPULATION SIZE = 100 ANDNUMBER OF GENERATIONS= 3000)

Function	**UF1**	**UF2**	**UF3**	**UF4**	**UF5**	**UF6**	**UF7**	**UF8**	**UF9**
Dying Rate in %	***SPREAD***								
10	0.67851	0.44045	**0.52668**	0.68793	0.79622	1.00287	0.78217	0.97061	0.71918
30	0.61903	0.46852	0.53844	0.61184	0.79091	0.91346	0.74607	0.91805	0.71540
40	0.59848	0.43489	0.56549	0.56923	0.79043	0.80435	0.72612	0.90348	0.69143
50	**0.58012**	**0.40815**	0.54745	**0.50465**	**0.67871**	**0.70065**	**0.67037**	**0.46400**	**0.63089**
60	0.59982	0.45439	0.52240	0.57528	0.74632	0.75061	0.80661	0.67061	0.65741
90	0.63690	0.43187	0.56346	0.56192	0.73732	0.84422	0.87310	0.78601	0.69461
	MeanIGD								
10	0.067327	0.023215	**0.201176**	**0.050215**	**0.153826**	**0.064901**	**0.083477**	0.864653	0.909465
30	0.068071	0.024151	0.210074	0.059157	0.183894	0.066933	0.103827	0.774346	**0.899907**
40	0.062276	0.024085	0.233127	0.062096	0.193835	0.069022	0.103612	0.760417	0.900172
50	**0.061657**	**0.019220**	0.240011	0.062340	0.214201	0.072117	0.098397	**0.755344**	0.914492
60	0.069975	0.021572	0.246741	0.063534	0.229206	0.078452	0.099014	0.774341	0.980227
90	0.063974	0.032322	0.256764	0.064323	0.246873	0.082166	0.133976	0.835970	0.940427

TABLE 5.2 PERFORMANCE OF PARREM2 ON UF1-UF9 (POPULATION SIZE= 100 ANDNUMBER OF GENERATIONS= 3000)

Function	**UF1**	**UF2**	**UF3**	**UF4**	**UF5**	**UF6**	**UF7**	**UF8**	**UF9**
Dying Rate in %	***MeanIGD***								
10	**0.060779**	0.024376	**0.170225**	0.0543763	0.208118	0.0704405	0.123257	0.840894	0.904564
30	0.062168	0.025683	0.188227	0.0543763	0.218096	0.0714245	0.113827	0.829490	0.919308
40	0.063844	0.026923	0.207625	0.0669232	0.219954	0.0710112	0.158997	0.804156	0.929701
50	0.066292	0.023877	0.200125	0.0694861	0.228448	0.0725229	0.143976	0.738952	0.934456
60	0.067125	0.025134	0.242268	0.0713947	0.241254	0.0734245	0.159071	0.760831	0.964970
90	0.065053	0.026968	0.251257	0.0709686	0.309919	0.0740860	0.161136	0.776082	1.089052
	SPREAD								
10	0.62275	0.48583	0.61062	0.71562	0.86114	0.82630	0.88275	0.72383	0.68716
30	0.60432	0.46012	0.51371	0.66981	0.85124	0.81630	0.87673	0.71085	0.58915
40	0.60432	0.46803	0.59376	0.60370	0.81002	0.82630	0.87856	0.70381	**0.58913**
50	**0.55935**	**0.45033**	**0.47090**	**0.53480**	**0.78428**	**0.76884**	**0.79031**	**0.68068**	0.64234
60	0.56134	0.46393	0.61360	0.75597	0.80242	0.84546	0.80663	0.73817	0.68061
90	0.56000	0.48436	0.55463	0.68868	0.82349	0.95473	0.93719	0.73134	0.68061

It is also observed that a moderate dying rate (50%) has given best spread for all functions. Fig 5.2 shows plot of SPREAD produced by ParRem1 for UF1-UF9. Fig 5.3 SPREAD produced by ParRem2 for UF1-UF9 with different Dying rates Fig 5.4 SPREAD produced by ParRem3 for UF1-UF9 with different Dying rates with different dying rates. Minimum value of SPREAD is reported with dying rate = 50%. Diversity among solutions deteriorates with increase in dying rate.

Strategy ParRem3: The Performance of dying strategy ParRem3 in terms of MeanIGD & SPREAD on function UF1-UF9 is shown in Table 5.3. The best MeanIGD value for all the function is obtained at low dying rate. Also as dying rate increases it increases MeanIGD value that reflects the degradation of performance of algorithm. Fig 5.4 shows plot of SPREAD produced by ParRem3 for UF1-UF9 with different dying rates. Best SPREAD on UF1 & UF5-UF7 is for dying rate =50%, function UF2-UF4 is for dying rate=30 and on function UF8 & UF9 is for dying rate = 10%.

TABLE 5.3 PERFORMANCE OF PARREM3 ON UF1-UF9 (POPULATION SIZE = 100 ANDNUMBER OF GENERATIONS = 3000)

Function	**UF1**	**UF2**	**UF3**	**UF4**	**UF5**	**UF6**	**UF7**	**UF8**	**UF9**
Dying Rate in %	***MeanIGD***								
10	**0.067689**	**0.022364**	**0.180125**	**0.054034**	**0.201256**	0.0596334	**0.110274**	0.777345	0.968565
30	0.068954	0.024135	0.189227	0.060135	0.201784	**0.0592211**	0.113827	**0.738637**	**0.959632**
40	0.068954	0.024813	0.209825	0.064815	0.201259	0.0609293	0.138097	0.756711	0.961038
50	0.069195	0.025761	0.210129	0.069316	0.234478	0.0642109	0.159442	0.836640	0.965698
60	0.070410	0.026186	0.230106	0.063186	0.238979	0.0662712	0.161913	0.922512	0.967616
90	0.074881	0.027356	0.259957	0.064156	0.247846	0.0672019	0.171136	0.932455	0.967229
	SPREAD								
10	0.58102	0.46043	0.54171	0.71257	0.93154	0.82297	0.80727	**0.65029**	**0.68203**
30	0.61351	**0.42707**	**0.50012**	**0.55244**	0.90016	0.82098	0.82407	0.68400	0.72314
40	0.56520	0.43101	0.50448	0.63289	0.82131	0.80232	0.85602	0.69864	0.73164
50	**0.55652**	0.43078	0.54705	0.63106	**0.63878**	**0.67963**	**0.70431**	0.83246	0.63450
60	0.67452	0.45257	0.50247	0.56174	0.68975	0.92213	0.76017	0.80253	0.72144
90	0.59006	0.45841	0.47691	0.57659	0.80108	0.96403	0.79309	0.78183	0.73340

5.5 Performance comparison of three strategies

Three strategies of dying of solution have been designed, implemented and seen their performance on function UF1-UF9. It is observed that for all three strategies best results for MeanIGD and SPREAD have been obtained for dying rate at or below 50%. Hence we can say that excessive dying of solutions affects the convergence to and diversity in Pareto front. Table 5.4 shows best performance (MeanIGD and SPREAD) of three strategies of dying of solution on function UF1-UF9. From Table 5.4 it is observed that dying strategy ParRem1 with dying rate 50% has outperformed ParRem2 and ParRem3 in terms of SPREAD on four functions (UF2, UF4, UF7 and UF8) out of nine functions. ParRem2 has outperformed ParRem1 and ParRem3 on three functions (UF1 & UF3 with dying rate 50% and UF9 with dying rate 40%). ParRem3 has outperformed ParRem1 and ParRem2 on two functions (UF5 and UF6 with dying rate 50%).

TABLE 5.4 COMPARISON OF PERFORMANCE OF PARREM1, PARREM2 AND PARREM3 ON FUNCTION UF1-UF9 WITH POPULATION SIZE = 100 AND NUMBER OF GENERATIONS = 3000

Scheme	**ParRem1**		**ParRem2**		**ParRem3**		***Sign test***
Function	***MeanIGD (Dying rate)***	***SPREAD (Dying rate)***	***MeanIGD (Dying rate)***	***SPREAD (Dying rate)***	***MeanIGD (Dying rate)***	***SPREAD (Dying rate)***	***(IGD, SPREAD)***
UF1	0.061657 (50%)	0.58012 (50%)	**0.060779** **10%)**	**0.55635** **(50%)**	0.067689 (10%)	0.55652 (50%)	**(- , -)**
UF2	**0.019220** **(50%)**	**0.40815** **(50%)**	0.023877 (50%)	0.45033 (50%)	0.022364 (10%)	0.42707 (30%)	**(+ , +)**
UF3	0.201176 (10%)	0.52668 (10%)	**0.170225** **10%)**	**0.47090** **(50%)**	0.180125 (10%)	0.50012 (30%)	**(- , -)**
UF4	**0.050215** **(10%)**	**0.50465** **(50%)**	0.054376 (10%)	0.53480 (50%)	0.054034 (10%)	0.55244 (30%)	**(+ , +)**
UF5	**0.153826** **(10%)**	0.67871 (50%)	0.208118 (10%)	0.78428 (50%)	0.201256 (10%)	**0.63878** **(50%)**	**(+ , -)**
UF6	0.064901 (10%)	0.70065 (50%)	0.070440 (10%)	0.76884 (50%)	**0.059221** **(30%)**	**0.67963** **(50%)**	**(- , -)**
UF7	**0.083477** **(10%)**	**0.67037** **(50%)**	0.113827 (30%)	0.79031 (50%)	0.110274 (10%)	0.70431 (50%)	**(+ , +)**
UF8	0.755344 (50%)	**0.46400** **(50%)**	0.738952 (50%)	0.68068 (50%)	**0.738637** **(30%)**	0.65029 (10%)	**(- , +)**
UF9	**0.899907** **(30%)**	0.63089 (50%)	0.904564 (10%)	**0.58913** **(40%)**	0.959632 (30%)	0.68203 (10%)	**(+ , -)**

TABLE 5.5 STATISTICAL SUM OF PROBLEMS FOR WHICH EACH STRATEGY OBTAINS SIGNIFICANTLY BETTER RESULTS

Metric	ParRem1	ParRem2	ParRem3
MeanIGD	05	02	02
SPREAD	04	03	02
Total	09	05	04

Table 5.5 shows Statistical Sum of Problems for which each strategy obtains significantly better results. ParRem1 has given statistically significant performance than ParRem2 and ParRem3 and hence dying strategy ParRem1 and dying rate 50% will be used for further experimentation.

Thus dying strategy ParRem1 is the best among the proposed three dying strategies and has helped 2 & 3-objective function having convex, non-convex, continuous and discontinuous Pareto fronts to converge to Pareto-optimal front and maintained diversity among solutions.

5.6 Comparison with other algorithms

The performance of proposed MOGA with dying strategy ParRem1 (MOGA-ParRem1) is compared with proposed MOGA without dying (MOGA-WD) and NSGA-II. To make a fair comparison, NSGA-II-MPX is coded in MatLab 7.1 and in place of SBX operator MPX crossover operator is used. Mutation operator is also removed from NSGA_II. Also parameter setting used for MOGA without dying and NSGA-II-MPX is same as given in section 4 except dying rate and 30 independent runs have been taken for each problem.

Table 5.6 shows MeanIGD and SPREAD of MOGA-WD, MOGA-ParRem1 and NSGA-II-MPX. SPREAD values reported in the table 5.6 indicate that MOGA-ParRem1 has outperformed MOGA-WD on functions UF1-UF9. Also MOGA-ParRem1 has outperformed NSGA-II-MPX by giving better SPREAD on functions UF1-UF4 and UF8-UF9. IGD measure taken for MOGA-ParRem1 shows that it has given good performance on functions UF1,UF2 and UF6 functions. NSGA-II-MPX has shown good IGD values on functions UF3-UF5 and UF7-UF9.Table 5.7 shows Statistical Sum of Problems for which each algorithm obtains significantly better results.

TABLE 5.6 COMPARISON OF PERFORMANCE OF MOGA-PARREM1 WITH MOGA AND NSGA-II ON FUNCTION UF1-UF9 (POPULATION SIZE = 100 AND NUMBER OF GENERATIONS = 3000)

Scheme	**MOGA-WD**		**MOGA-ParRem1 (DR = 50%)**		**NSGA-II-MPX**		***Sign test***
Function	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***	***(MeanIGD, SPREAD)***
UF1	0.061892	0.615871	**0.061657**	**0.580122**	0.068187	0.697345	(+ , +)
UF2	0.057764	0.644613	**0.019220**	**0.408153**	0.024311	0.540971	(+ , +)
UF3	0.245993	0.583918	0.240011	**0.547451**	**0.135773**	0.666309	(- , +)
UF4	0.066059	0.550369	0.062340	**0.504653**	**0.019737**	0.682081	(- , +)
UF5	1.046338	0.753780	0.214201	0.678715	**0.211415**	**0.658579**	(- , -)
UF6	0.230648	0.695138	**0.072117**	0.700656	0.152526	**0.694642**	(+, -)
UF7	0.101259	0.715222	0.098397	0.670374	**0.031356**	**0.626167**	(- , -)
UF8	0.836721	0.686384	0.755344	**0.464006**	**0.536727**	0.616086	(- , +)
UF9	0.975251	0.883306	0.914492	**0.630897**	**0.375251**	0.703306	(- , +)

Table 5.6 shows MeanIGD and SPREAD of MOGA-WD, MOGA-ParRem1 and NSGA-II-MPX. SPREAD values reported in the table 5.6 indicate that MOGA-ParRem1 has outperformed MOGA-WD on functions UF1-UF9. Also MOGA-ParRem1 has outperformed NSGA-II-MPX by giving better SPREAD on functions UF1-UF4 and UF8-UF9. IGD measure taken for MOGA-ParRem1 shows that it has given good performance on functions UF1,UF2 and UF6 functions. NSGA-II-MPX has shown good IGD values on functions UF3-UF5 and UF7-UF9.Table 5.7 shows Statistical Sum of Problems for which each algorithm obtains significantly better results.

TABLE 5.7 STATISTICAL SUM OF PROBLEMS FOR WHICH EACH ALGORITHM OBTAINS SIGNIFICANTLY BETTER RESULTS

Metric	**MOGA-WD**	**MOGA-ParRem1 (DR = 50%**	**NSGA-II-MPX**
MeanIGD	0	03	06
SPREAD	0	06	03
Total	0	09	09

TABLE 5.8 COMPARISON OF CPU TIME OF MOGA-PARREM1 WITH MOGA AND NSGA-II ON FUNCTION UF1-UF9 (POPULATION SIZE = 100 AND NUMBER OF GENERATIONS = 3000)

Function	**MOGA-WD**	**MOGA-ParRem1 (DR = 50%**	**NSGA-II-MPX**
UF1	**162.56**	169.23	1008.341
UF2	**148.32**	192.11	1014.821
UF3	**172.35**	198.42	1028.630
UF4	**146.71**	174.54	1002.473
UF5	**181.62**	205.71	1068.459
UF6	**175.34**	186.34	1015.378
UF7	**144.37**	197.19	1013.489
UF8	**180.34**	220.16	1104.672
UF9	**192.45**	234.03	1100.592

Next we consider simulation speed as metric for performance measure. Table 5.8 shows the CPU time (in seconds) taken by the MOGA-WD, MOGA-ParRem1 and NSGA-II-MPX on problems UF1-UF9. Strategy ParRem1 have complexity of O(N^2) because in order to remove a solution it has to find solution having minimum distance from other solution where N is number of solutions. . For NSGA-II, the complexity to obtain the overall non-dominated set is O (MN^2). Complexity of MOGA-WD is O(N) where M is number of objectives and N is size of population. MOGA-WD reports minimum CPU time for UF1-UF9 and MOGA-ParRem1 requires little bit more CPU time for UF1-UF9 as compared to MOGA-WD. Reason for increase in CPU time is the complexity O(N^2) of strategy ParRem1. For UF1-UF9 highest CPU time is reported for NSGA-II-MPX. This is because of complexity O (MN^2) of Non-dominated sorting technique and NSGA-II has to execute Non-dominated sorting in every generation whereas MOGA-ParRem1 has to run strategy ParRem1 after Gen_Rem generation only. Number of generations in which MOGA-ParRem1 executes ParRem1 is very-very less than maximum number of generations.

Fig 5.5-5.13 shows plots of non dominated set of solutions obtained on functions UF2, UF4 and UF7 with MOGA-WD, MOGA-ParRem1 and NSGA-II-MPX. From fig. 5.6 & 5.9 it is clear that MOGA-ParRem1 has given better diversity and convergence than MOGA-WD and NSGA-II-MPX on test functions UF2 &

UF4. Poor performance of MOGA-ParRem1 is seen in fig. 5.12 on function UF7. MOGA-WD has shown worst performance on all three test functions as shown in fig. 5.5, 5.8 & 5.11.

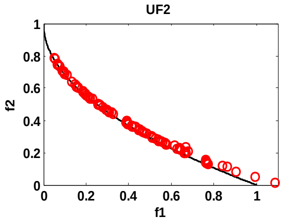

Fig. 5.5 Best approximate of UF2 with MOGA-WD

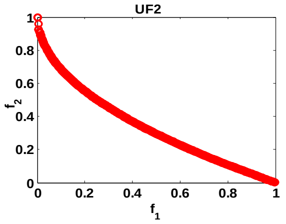

Fig. 5.6 Best approximate of UF2 with MOGA-ParRem1

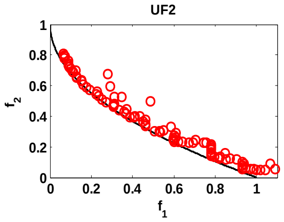

Fig. 5.7 Best approximate of UF2 with NSGA-II-MPX

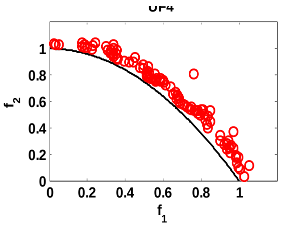

Fig. 5.8 Best approximate of UF4 with MOGA-WD

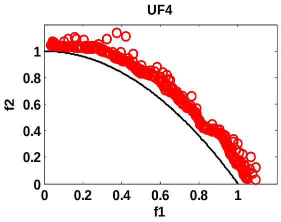

Fig. 5.9 Best approximate of UF4 with MOGA-ParRem1

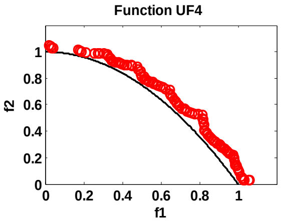

Fig. 5.10 Best approximate of UF4 with NSGA-II-MPX

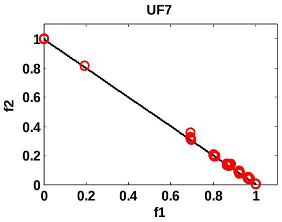

Fig. 5.11 Best approximate of UF7 with MOGA-WD

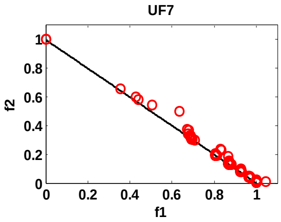

Fig. 5.12 Best approximate of UF7 with MOGA-ParRem1

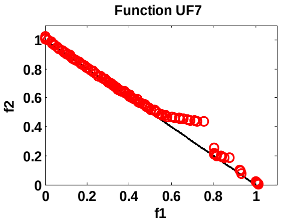

Fig.5.13 Best approximate of UF7 with NSGA-II-MPX

Table 5.9 presents computational results of SC metric for all test algorithms. NSGA-II-MPX has outperformed MOGA-WD on all test functions by giving value of SC metric nearly equal to one. When relative coverage comparison is done between MOGA-ParRem1 and MOGA-WD, it is found that MOGA-ParRem1 has given better values of SC metric over MOGA-WD for all test functions. Non dominated set of MOGA-ParRem1 is compared with non dominated set of NSGA-II-MPX and relative set coverage of two sets shows that MOGA-ParRem1 is better than NSGA-II-MPX on UF1-UF4 & UF8,UF9 test functions whereas NSGA-II-MPX has outperformed MOGA-ParRem1 on test functions UF5-UF7.

TABLE 5.9 TWO SET COVERAGE METRIC *SC*

Function	X	(x,MOGA-WD)	(x , MOGA-ParRem1)	(x , NSGA-II-MPX)
UF1	MOGA-WD	--	0.3400	0.0
	MOGA-ParRem1	0.9600	--	0.9000
	NSGA-II-MPX	0.9900	0.4200	--
UF2	MOGA-WD	-	0.3000	0.0
	MOGA-ParRem1	0.9500	--	0.9300
	NSGA-II-MPX	0.9900	0.4700	--
UF3	MOGA-WD	--	0.2700	0.1000
	MOGA-ParRem1	0.9000	--	0.6500
	NSGA-II-MPX	0.9800	0.5300	--
UF4	MOGA-WD	--	0.1900	0.0900
	MOGA-ParRem1	0.9900	--	0.9700
	NSGA-II-MPX	0.9900	0.4900	--
UF5	MOGA-WD	-	0.3400	0.1200
	MOGA-ParRem1	0.8900	-	0.5600
	NSGA-II-MPX	0.9900	0.7800	-
UF6	MOGA-WD	-	0.3000	0.2000
	MOGA-ParRem1	0.8200	-	0.5500
	NSGA-II-MPX	0.9700	0.6500	-
UF7	MOGA-WD	-	0.2100	0.1000
	MOGA-ParRem1	0.9200	-	0.3500
	NSGA-II-MPX	0.9900	0.8600	-
UF8	MOGA-WD	-	0.2000	0.1000
	MOGA-ParRem1	0.9100	-	0.8400
	NSGA-II-MPX	0.9700	0.3000	-
UF9	MOGA-WD	-	0.2600	0.1200
	MOGA-ParRem1	0.9400	-	0.8600
	NSGA-II-MPX	0.9400	0.3800	-

5.6 Key findings

In this study GA procedure is made more close to natural evolutionary process which incorporate Birth-Reproduction-death cycle. The major contribution of this study is the introduction of dying strategy as explicit part of evolutionary process. This chapter has given insight into dying of solutions and its impact on diversity of population. Dying is implicit part of selection process i.e. solutions which are not selected die out. In this chapter proposed and implemented dying strategies are presented. Three strategies for dying of solution from next generation population have been incorporated in a new MOGA framework. Performances of these strategies have been tested on test functions of IEEECEC09 Test Suit.

The empirical study suggest the following

- Proposed MOGA with three strategies of dying of solutions is able to guide the search process towards the optimum for the seven bi-objective and the two 3-objective test functions.
- It is found that Inverted Generational Distance (IGD) increases with increase in dying rate of solutions.
- Also, as dying rate increases diversity among solution increases but diversity deteriorates when dying rate exceeds 50%. So we can conclude that a moderate dying rate (around 50% or less than 50%) gives the better spread.

A comparative analysis of three dying strategies is also done in this study. On comparing performance of three strategies of dying of solutions we found that:

- Among the three strategies of dying of solutions, strategy ParRem1 has given statistically significant results on 4 functions (UF2, UF4, UF7 and UF8) in terms of SPREAD metric. These functions are continues, convex and non convex functions
- ParRem2 and ParRem3 have given statistically significant performance on three (UF1, UF3 & UF9) (Convex & disconnected Pareto front) and two (UF5 & UF6) (Discontinues) functions respectively

From the above observation we conclude that, ***“different strategies work well for different problems”.***

When MOGA-ParRem1 compared with MOGA-WD it is found that MOGA-ParRem1 has outperformed MOGA-WD on all functions by giving better diversity. MOGA-ParRem1 has also outperformed NSGA-II-MPX on many functions in terms of SPREAD, IGD and SC metric. Also Moga-ParRem1 is faster than NSGA-II-MPX.

Thus study has successfully shown that MOGA with explicit dying mechanism with controlled dying rate can improve performance of genetic algorithm. And hence it is concluded that dying rate of solution has some impact on performance of GA.

Chapter 6

Ensemble of Dying Strategy based MOGA

Excessive dying in nature causes reduction in diversity and leads to extinction of organism. In previous chapter three explicit dying strategies are presented. Using dying strategy a solution is removed from next generation population in a deterministic way. MOGA takes decision of removal of solution, based on one of these three strategies. MOGA with dying strategy is tested on nine functions (UF1-UF9). Empirically it is found that Among the three strategies of dying, strategy ParRem1 has given statistically significant results on 4 functions (UF2, UF4, UF7 & UF8), ParRem2 and ParRem3 have given statistically significant performance on three (UF1, UF3 & UF9) and two (UF5&UF6) functions respectively and hence it is concluded that no single strategy have got successes on all the functions. In this study to improve performance of MOGA ensemble learning is used and an ensemble of dying strategy based MOGA is presented.

6.1 Introduction

Genetic Algorithm (GA) mimics the process of natural selection and is robust tool for search and optimization problems [1]. Many researchers have proven this. Suganthan et.al in [91], present the development of multi-objective evolutionary algorithms (MOEAs) primarily, during the last eight years. The survey covers algorithmic frameworks such as decomposition-based MOEAs (MOEA/Ds), memetic MOEAs, co-evolutionary MOEAs, selection and offspring reproduction operators, MOEAs with specific search methods, MOEAs for multimodal problems, constraint handling and MOEAs, computationally expensive multi-objective optimization problems (MOPs), dynamic MOPs, noisy MOPs, combinatorial and discrete MOPs, benchmark problems, performance indicators, and applications. In addition, some future research issues are also presented.

Genetic algorithm uses selection, crossover and mutation operators to evolve a set of solutions of current generation. Selection operator is then used to select fit solutions from current generation as next generation population [1]. Weaker

Solutions which are not selected die out and do not reappear. Thus dying is implicit part of selection mechanism. Solutions having better fitness produce fitter offspring and selection strategies are likely to select both parent and their offspring. Offspring solutions inherits the properties of Solutions (parent solution) present in first generation using cross over operator. Offspring solutions generated using crossover operator carry forward the good properties of parent solutions in subsequent generations. Problem with this selection strategy is that, after few generation whole population is dominated by presence of few solutions from initial population and their offspring i.e. trail of very few solutions from initial population reach to final generation and most of the solutions die out somewhere in between generations. Thus selection strategy based on survival of fittest reduces diversity and convergence of solutions. So, a new mechanism is needed to avoid excessive dying of solutions.

In the study presented in previous chapter dying has been made explicit part of evolutionary process to control excessive dying of solutions, and three strategies for dying of solution have been proposed and implemented. Using one of the three proposed dying strategies a solution is deterministically removed from next generation population. Impact of dying rate of solutions on the performance of GA has been studied. Idea of gradual dying is modeled and a new framework of MOGA has been used to demonstrate the same. Experimental evidences show that any one strategy is not sufficient to solve all the problems.

Given an optimization problem (i.e. objective function) *f* and an algorithm *a*, it is important to have some measure of how well *a* performs on *f*. Moreover, given empirical evidence of *a*’s performance on *f*, is it possible to make generalizations about *a*’s performance on other functions, either of the same or different type as *f*? Intuition would have one believe that there are some algorithms that will perform better than others on average. However, the No Free Lunch (NFL) Theorems state that such an assertion cannot be made. That is, across all optimization functions, the average performance of all algorithms is the same. This means that if an algorithm performs well on one set of problems then it will perform poorly (worse than random search) on all others [92].

From the empirical results presented in previous chapter it is concluded that different dying strategies have given good performance on different MOPs. Hence there is need to design a MOGA which can select appropriate strategy among the three dying strategies for an MOP. Using ensemble learning technique a MOGA with three dying strategies can be designed. Ensemble Learning refers to the procedures employed to train multiple learning machines and combine their outputs, treating them as a committee of decision makers. The principle is that the committee decision, with individual predictions combined appropriately, should have better overall accuracy, on average, than any individual committee member. Numerous empirical and theoretical studies have demonstrated that ensemble models very often attain higher accuracy than single models. The members of the ensemble might be predicting real-valued numbers, class labels, posterior probabilities, rankings, clustering, or any other quantity. Therefore, their decisions can be combined using many methods, including averaging, voting, and probabilistic methods. The majority of ensemble learning methods are generic, applicable across broad classes of model types and learning tasks.

This chapter introduces ensemble of dying strategy based MOGA (EDS-MOGA) that can optimize given set of problems. In EDS-MOGA, ensemble of three dying strategy is formed and population is evaluated by selecting strategy one by one from the ensemble of dying strategies for few generations. Performance of each strategy is evaluated and best strategy is selected for rest of the generations.

6.2 Ensemble learning and Evolutionary algorithms

Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem. An ensemble combines a series of k learned models D1, D2,…,Dk, with the aim of creating an improved composite model D*. A given set of solutions P is used to create k sets, P1, P2,…Pk where Di ($1<i<=k-1$) is used to solve Pi. Ensemble is able to boost a weak learner to strong learner. There are many effective ensemble methods. The three representative methods are *Boosting*, *Bagging* and *Stacking*. [101]

The Mixtures of Experts architecture is a widely investigated paradigm for creating an ensemble [102]. The principle underlying the architecture is that certain models will be able to `specialize' to particular parts of the input space. It is commonly implemented with a Neural Network as the base model, or some other model capable of estimating probabilities. A Gating network receives the same inputs as the component models, but its outputs are used as the weights for a linear combiner. The Gating network is responsible for learning the appropriate weighted combination of the specialized models (or experts) for any given input. In this way the input space is `carved-up' between the experts, increasing and decreasing their weights for particular examples. In effect, a Mixture of Experts explicitly learns how to create expert ensemble members in different portions of the input space, and select the most appropriate subset for a new testing example.

Ensemble learning has already been used in diverse applications such as optical character recognition, text categorization, face recognition, computer-aided medical diagnosis, gene expression analysis, etc. Actually, ensemble learning can be used wherever machine learning techniques can be used.

Ensemble learning has proven to be very efficient and effective for adjusting algorithmic control parameters and operators in an online manner. Reference [93] shows the use of an ensemble of different Neighborhood Sizes, in MOEA/D and the neighborhood sizes dynamically adjust their selection probabilities based on their previous performances. In the paper [98], differential evolution with an ensemble of restricted tournament selection (ERTS-DE) algorithm is introduced to perform multimodal optimization. It is impossible for a single constraint handling technique to outperform all other constraint handling techniques always on every problem irrespective of the exhaustiveness of parameter tuning. To overcome this selection problem, an ensemble of constraint handling methods (ECHM) to tackle constrained multi-objective optimization problems has been used. The ECHM is integrated with multi-objective differential evolution (MODE) algorithm [97]**.** In [99] authors propose an ensemble of mutation and crossover strategies and parameter values for DE (EPSDE) in which a pool of mutation strategies, along with a pool of values corresponding to each associated parameter competes to produce successful offspring population. Thus the work done in [93]-[99] motivated authors to use ensemble methods to improve performance of MOGA with dying strategy.

6.3 Ensemble of Dying strategy based MOGA

In this section EDS-MOGA is presented. This is a novel MOGA with ensemble of dying strategies. Concept of ensemble learning is used to combine the three strategies in a MOGA. learning is presented. This algorithm uses mixture of experts' architecture is used to collaborate three dying strategies. In the beginning each initial solution is given a unique number as *parent number*. The algorithm divides input population into three subpopulations. It maintains a pool of three dying strategies. A Learning Period (LP) is defined. Parameter values for max gen, DR, Gen_Rem and iGen_Rem is set. Each subpopulation is randomly assigned a dying strategy from the pool. Each subpopulation undergoes cycles of selection-crossover-multilevel tournament selection. Using corresponding dying strategy a solution is removed from each subpopulation in a deterministic way. Subpopulations evolve independently for LP number of generations. After LPth generation performance of each subpopulation is evaluated. Here improvement in objective values is the criteria used for performance comparison of the three strategies. The dying strategy assigned to subpopulation showing large improvement in objective values is selected as winner dying strategy. All the three subpopulations are combined and evaluated with winner dying strategy for rest of the generation. Flowchart of proposed Ensemble of Dying Strategy Based Multi-Objective Genetic Algorithm is given in fig. 6.1.

6.4 Experimental design and results

Experiments are performed to demonstrate the behavior of the proposed approach in terms of convergence and diversity. Dying rate is chosen as 50% and other parameter setting used are same as given in chapter 5, section 5.4.

TABLE 6.1 PERFORMANCE COMPARISON OF EDS-MOGA WITH ParRem1, ParRem2 AND ParRem3

Scheme	**ParRem1**		**ParRem2**		**ParRem3**		**EDS-MOGA**	
Function	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***
UF1	0.061657	0.58012	0.060779	0.55635	0.067689	0.55652	**0.054910**	0.55891
UF2	0.019220	0.40815	0.023877	0.45033	0.022364	0.42707	**0.013663**	**0.40647**
UF3	0.201176	0.52668	0.170225	0.47090	0.180125	0.50012	**0.097539**	0.49136
UF4	0.050215	0.50465	0.054376	0.53480	0.054034	0.55244	**0.016117**	0.50529
UF5	0.153826	0.67871	0.208118	0.78428	0.201256	0.63878	**0.094909**	**0.53358**
UF6	0.064901	0.70065	0.070440	0.76884	0.059221	0.67963	0.060966	**0.66912**
UF7	0.083477	0.67037	0.113827	0.79031	0.110274	0.70431	**0.080293**	**0.66937**
UF8	0.755344	0.46400	0.738952	0.68068	0.738637	0.65029	**0.271814**	**0.46076**
UF9	0.899907	0.63089	0.904564	0.58913	0.959632	0.68203	**0.165976**	**0.55382**

MeanIGD and SPREAD values of UF1-UF9 produced by EDS-MOGA are reported in table 6.1. Table 6.1 also shows MeanIGD and SPREAD values of three dying strategies on UF1-UF9 function. EDS-MOGA has outperformed MOGA with single dying strategy i.e.ParRem1, ParRem2 and ParRem3 on functions UF1-UF5 & UF7-UF9 in terms of IGD metric and on functions UF2, UF5-UF9 functions in terms of SPREAD metric. EDS-MOGA has comparable performance in other cases. EDS-MOGA combines' features of three dying strategy and contribution of each strategy has produced good convergence and diversity in the population. EDS-MOGA dynamically selects the dying strategies. The selection of appropriate strategy by EDS_MOGA has helped in improving performance of EDS-MOGA

Among the three dying strategies, strategy ParRem1 has performed better than the other two strategies. Hence ParRem1 is the winner dying strategy. To show that EDS-MOGA is better than MOGA-ParRem1 we plot nondominated sets obtained by two algorithms on UF1-UF9 functions. Fig 6.2-6.10 shows plots of nondominated sets of functions UF1-UF9 obtained by MOGA-ParRem1 and fig 6.11-6.19 shows plots of nondominated sets of functions UF1-UF9 obtained by EDS-MOGA.

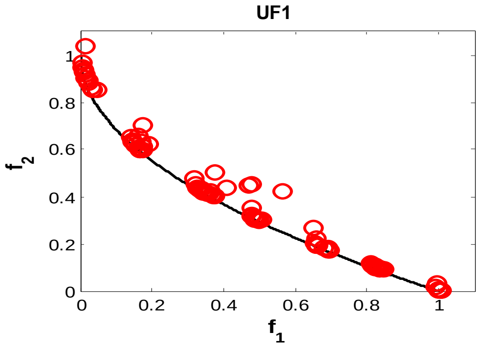

Fig 6.2 Nondominated set obtained by MOGA-ParRem1 on UF1

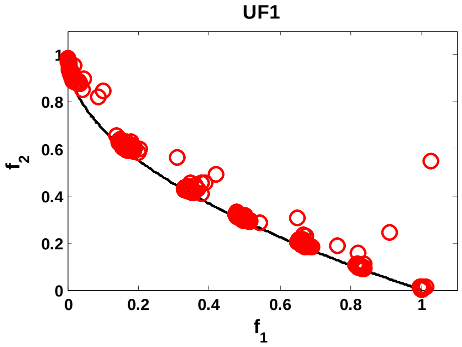

Fig 6.11 Nondominated set obtained by EDS-MOGA on UF1

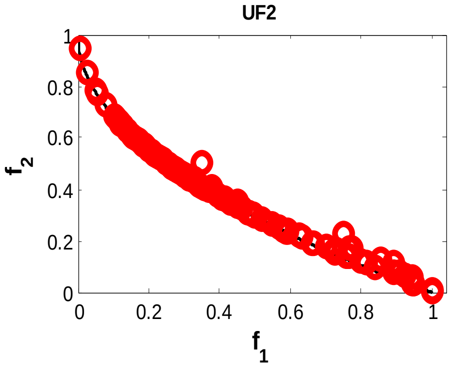

Fig 6.3 Nondominated set obtained by MOGA-ParRem1 on UF2

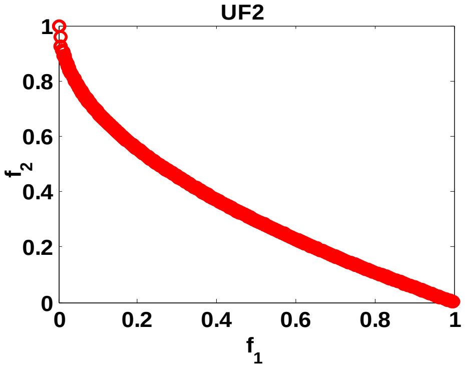

Fig 6.12 Nondominated set obtained by EDS-MOGA on UF2

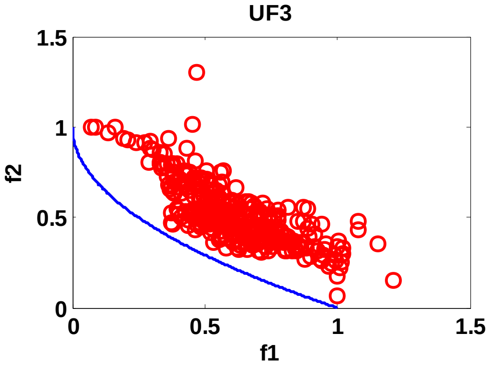

Fig 6.4 Nondominated set obtained by MOGA-ParRem1 on UF3

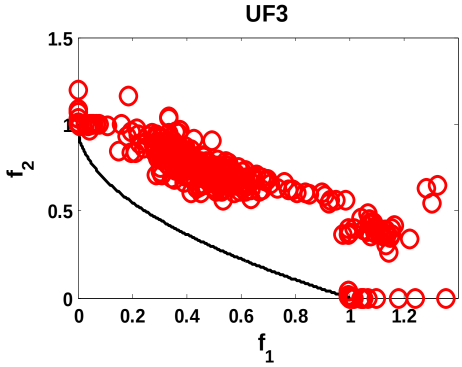

Fig 6.13 Nondominated set obtained by EDS-MOGA on UF3

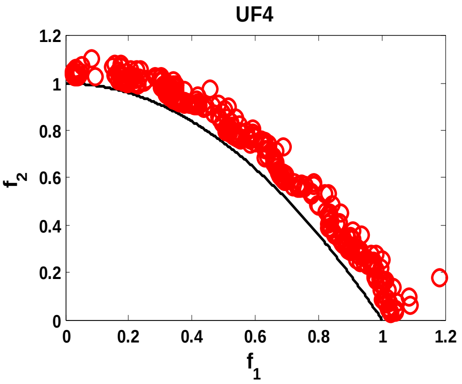

Fig 6.5 Nondominated set obtained by MOGA-ParRem1 on UF4

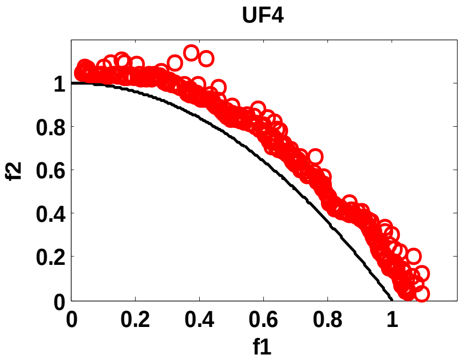

Fig 6.14 Nondominated set obtained by EDS-MOGA on UF4

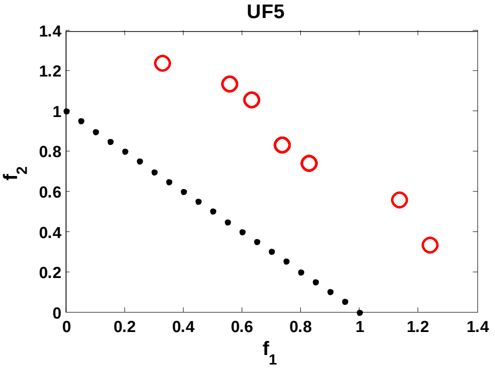

Fig 6.6 Nondominated set obtained by MOGA-ParRem1 on UF5

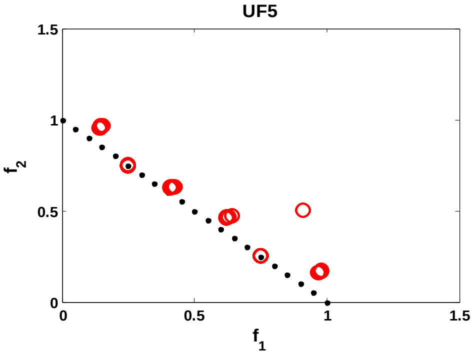

Fig 6.15 Nondominated set obtained by EDS-MOGA on UF5

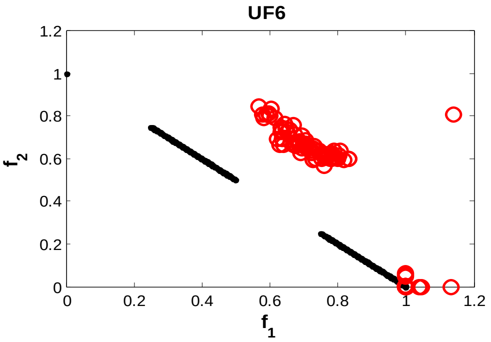

Fig 6.7 Nondominated set obtained by MOGA-ParRem1 on UF6

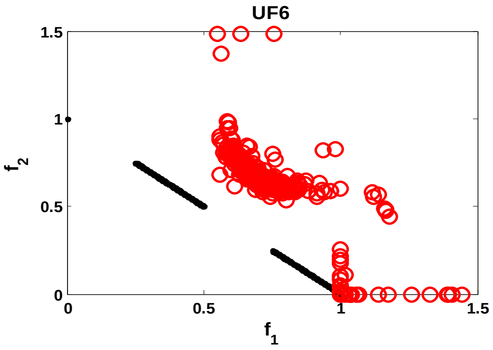

Fig 6.16 Nondominated set obtained by EDS-MOGA on UF6

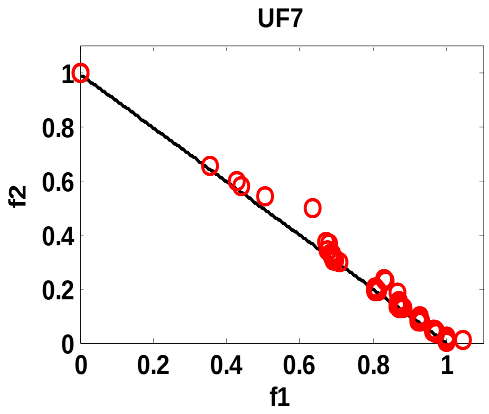

Fig 6.8 Nondominated set obtained by MOGA-ParRem1 on UF7

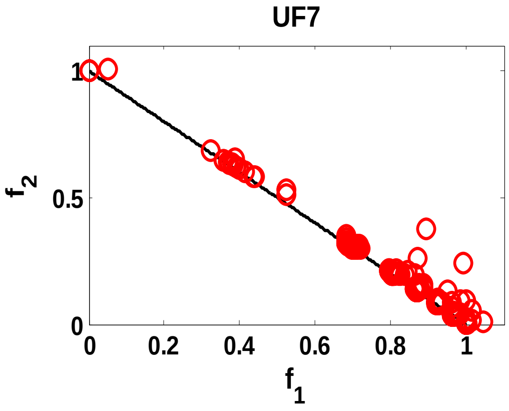

Fig 6.17 Nondominated set obtained by EDS-MOGA on UF7

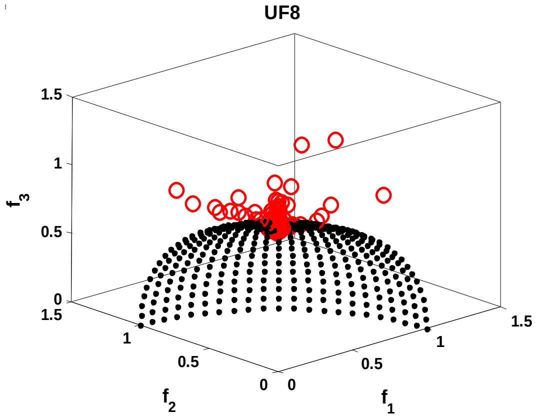

Fig 6.9 Nondominated set obtained by MOGA-ParRem1 on UF8

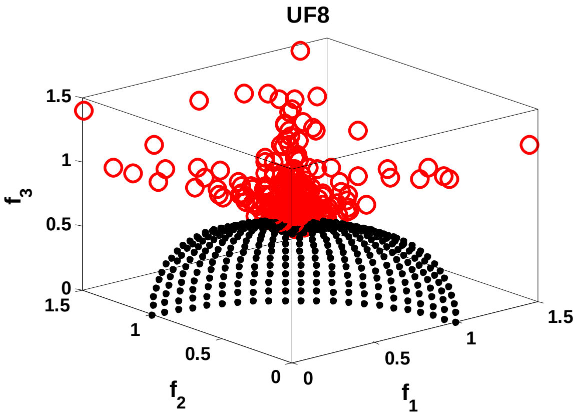

Fig 6.18 Nondominated set obtained by EDS-MOGA on UF8

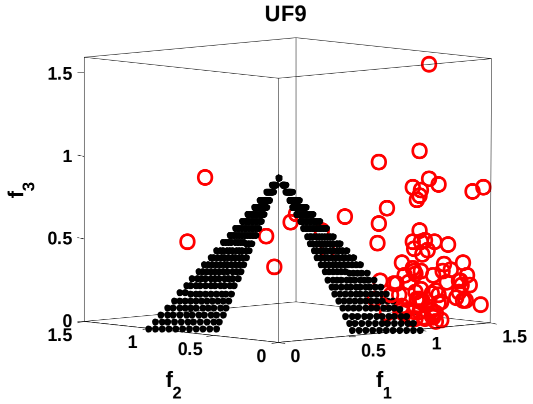

Fig 6.10 Nondominated set obtained by MOGA-ParRem1 on UF9

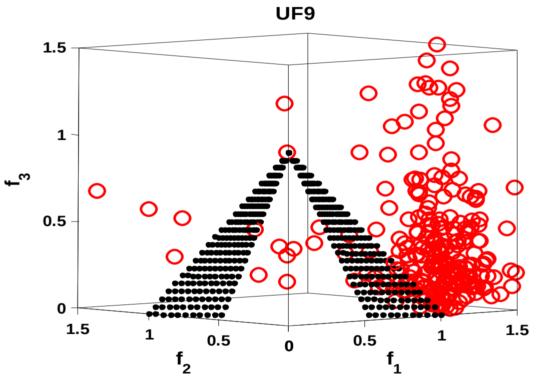

Fig 6.19 Nondominated set obtained by EDS-MOGA on UF9

6.5 Performance comparison with other algorithm

Performance of EDS-MOGA is compared with NSGA-II and SNOVMOGA algorithms. For fair comparison SBX operator in NSGA-II is replaced by MPX operator and mutation operator is also removed. NSGA-II-MPX is coded in MatLab 7.1. SNOVMOGA uses Summation of normalized objective value based sorting as discrimination technique while selecting parent solutions for formation of next generation population. In order to have diversity among solutions it uses reference point based improved selection method.

MeanIGD and SPREAD values of EDS-MOGA, NSGA-II-MPX and SNOVMOGA algorithms are reported in Table 6.2. EDS-MOGA has outperformed NSGA-II-MPX on functions UF1-UF6 &UF8-UF9. NSGA-II-MPX has given better MeanIGD and SNOVMOGA has given better SPREAD on UF7 function when compared to EDS-MOGA. Also SNOVMOGA has outperformed EDS-MOGA on function UF1 and UF7 in terms of SPREAD metric.

TABLE 6.2 PERFORMANCE COMPARISON OF EDS-MOGA WITH NSGA-II-MPX AND SNOVMOGA

Scheme	**EDS-MOGA**		**NSGA-II-MPX**		**SNOVMOGA**	
Function	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***
UF1	**0.054910**	0.55891	0.068187	0.697345	0.057391	**0.536142**
UF2	**0.010663**	**0.40647**	0.024311	0.540971	**0.011047**	0.428002
UF3	**0.097539**	**0.52136**	0.135773	0.666309	**0.031120**	0.588970
UF4	**0.016117**	**0.50529**	0.019737	0.682081	**0.016262**	0.631167
UF5	**0.094909**	**0.53358**	0.211415	0.658579	0.819614	0.697120
UF6	**0.029566**	**0.66912**	0.152526	0.694642	0.311660	0.684705
UF7	0.080293	0.65037	**0.031356**	0.606167	**0.010428**	**0.531024**
UF8	**0.271814**	**0.46076**	0.536727	0.616086	**0.322575**	0.793246
UF9	0.125976	**0.55382**	0.375251	0.703306	**0.108460**	0.604505

Fig 6.20-6.28 shows plots of nondominated set obtained by EDS-MOGA and NSGA-II-MPX. Function UF5 is having discontinues Pareto-front. EDS-MOGA has successfully converged solutions on the Pareto-front of UF5 function whereas NSGA-II-MPX fails on UF5 as solutions are away from Pareto-front. EDS-MOGA

has given poor spread on function UF7 as shown in fig. 6.6 where as NSGA-II-MPX has given good spread on UF7 as shown in fig.6.7. EDS-MOGA has shown better convergence and diversity when compared to NSGA-II-MPX.

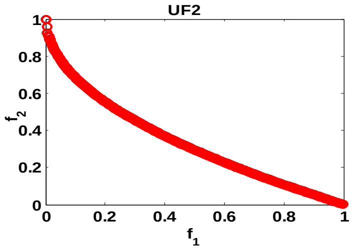

Figure 1. Fig.6. 20 Best approximate of UF2 with EDS-MOGA

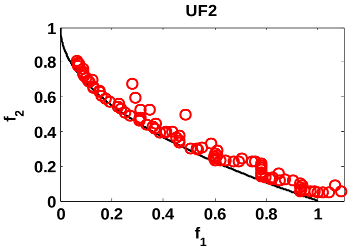

Fig. 6.24 Best approximate of UF2 with NSGA-II-MPX

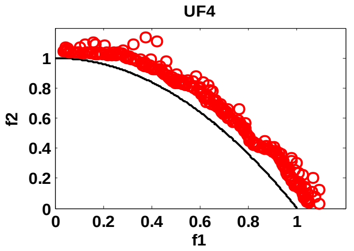

Figure 2. Fig. 6.21 Best approximate of UF4 with EDS-MOGA

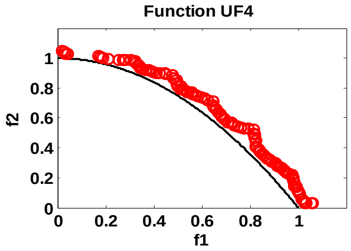

Figure 3. Fig. 6.25 Best approximate of UF4 with NSGA-II-MPX

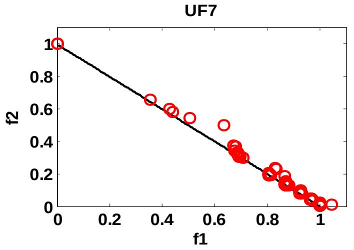

Figure 4. Fig. 6.22 Best approximate of UF7 with EDS-MOGA

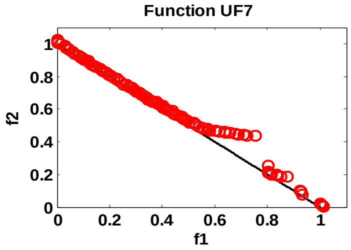

Figure 5. Fig 6.26 Best approximate of UF7 with NSGA-II-MPX

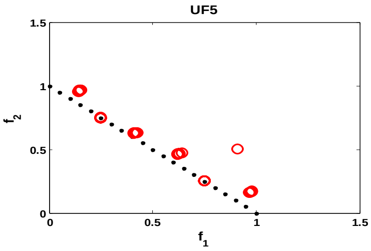

Figure 6. Fig. 6.23 Best approximate of UF5 with EDS-MOGA

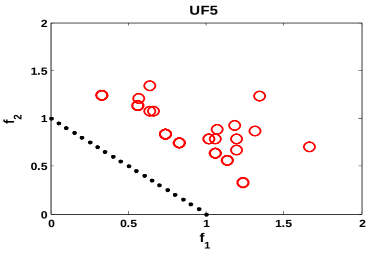

Fig.6.27 Best approximate of UF5 with NSGA-II-MPX

Performance comparison of EDS-MOGA with MOEA/D on UF5 function is shown in table 6.3.

TABLE 6.3 PERFORMANCE OF MOEA/D AND EDS-MOGA ON UF5 FUNCTION

SCHEME	MEANIGD	STDIGD	MINIGD	MAXIGD
MOEA/D	0.180711	0.068112	0.080283	0.306214
EDS-MOGA	**0.094909**	**0.018956**	**0.039203**	0.574952

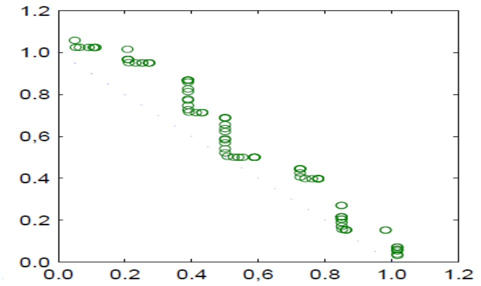

Figure 2. Fig. 6.28a Best approximate of UF5 with MOGA/D

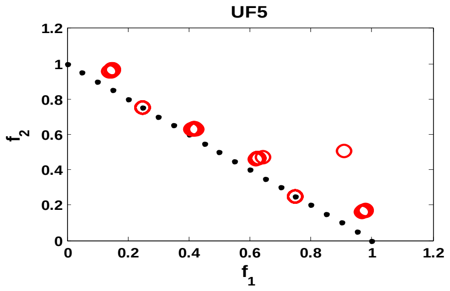

Figure 1. Fig. 6.28b Best approximate of UF5 with EDS MOGA

EDS-MOGA has outperformed MOEA/D on UF5 function. UF5 is a difficult function with disconnected and uniform Pareto front. EDS-MOGA has remarkably good performance on UF5 function. Plots of nondominated solutions obtained by the two algorithms on UF5 function are shown in fig. 6.28.

6.6 Key findings

Previous chapter has presented three dying strategies, their performance on different functions. From the empirical results it is concluded that one single dying strategy has capability to solve all nine problems (UF1-UF9).

In this chapter a novel MOGA has been presented. EDS-MOGA uses ensemble learning technique to combine the three dying strategies in a single MOGA. A brief introduction to ensemble learning and its foot prints in evolutionary algorithms also given in the chapter. EDS-MOGA uses all the three strategies of dying to evaluate solutions for few generations and then selects one dying strategy i.e. best dying strategy for rest of the generations. Selection of dying strategy is based on performance of strategy in learning Period (LP) generations.

EDS-MOGA has outperformed MOGA with single dying strategy. Also EDS-MOGA has shown better performance on many functions when compared to NSGA-II-MPX and SNOVMOGA.EDS-MOGA has given remarkably good performance on UF5 functions whereas all the MOGAs proposed in this study have performed poorly on UF5. In the LP generation all the three dying strategies contribute in evolution process and after checking performance of all the strategies algorithm selects best strategy for evolution. This effort of collaboration of three strategy in initial generation has helped solutions to have convergence and diversity and thus helped in the improvement of performance of EDS-MOGA. Thus ensemble of dying strategy has improved performance of MOGA.

Chapter 7

Results and Conclusions

The research work is aimed at the performance enhancement of Real-Coded Genetic Algorithms used for multi-objective optimization. Study started with identifying the potential areas in RCGAs where modification might enhance their performance. Operators play very important role in the functioning of GAs. In GAs the roles of selection and recombination (or crossover) operators are very well defined. Selection operator controls the direction of a search and the recombination operator generates new vistas for the search. The efficacy of GAs on a particular problem hinges quite strongly on the degree of exploration and exploitation of search space by the recombination operator and the direction of search in the search space set by the selection operator. GA uses simple selection operators for solving single objective optimization problems but special selection operators are needed to solve multi-objective optimization problems. Study proposes several selection schemes and special scheme called dying strategy and used them in new MOGA framework.

7.1 Summary

In chapter 2 an insight is given on Pareto-dominance based selection scheme and crossover operator used in NSGA-II. NSGA-II is modified and SBX operator is replaced by MPX and MLX operators. Performance of MPX and MLX operators is investigated on few functions from ZDT test suit. It is found that found that NSGA-II-MPX and NSGA-II-MLX has given better performance.

In chapter 3, two non-Pareto based selection approaches, Rank-sum and Summation of Normalized Objective Value as an alternative to Pareto-dominance based selection have been reviewed. Two multi-objective Genetic algorithm RSMOGA and SNOVMOGA based on above-mentioned selection approaches have been implemented and tested on unconstrained test problems of CEC09 test suit. Experiments have been performed to investigate the search capability of these algorithms and the effectiveness of the two approaches in handling the issues of convergence and diversity in multi-objective optimization problems. Also the two methods have been compared with Pareto-dominance based non-dominated sorting technique (NSGA-II-MPX) as shown in table 7.1.

Experimental results indicate that RSMOGA and SNOVMOGA algorithms are able to guide the search process towards the optimum for the seven bi-objective and the two 3-objective test functions. SNOV sort technique is fine-grained ranking technique and provides high order ranking and has helped the SNOVMOGA to give better convergence and diversity in comparison to RSMOGA and NSGA-II-MPX. Sign test is also performed and shown in last column in table 7.1. A + indicate good or comparable performance and – indicate insignificant performance. Table 7.2 shows that SNOVMOGA has given statistically significant performance on seven functions in terms of MeanIGD metric and on six functions in terms of SPREAD metric. On the other hand RSMOGA has shown better computational efficiency on all nine functions. SNOVMOGA and NSGA-II-MPX have shown poor efficiency. Rank-sum sort is less complex technique and adaptive diversified selection requires less iteration for selection of solutions for next generation.

From the above discussion it is clear that SNOVMOGA has shown better convergence and diversity in all test functions having continuous convex and non-convex Pareto fronts but performed poorly on functions having discontinuous Pareto fronts.

TABLE 7.1 COMPARISON OF SNOVMOGA WITH RSMOGA & NSGA-II-MPX

Scheme	**NSGA-II-MPX**			**RSMOGA**			**SNOVMOGA**			**Sign test**
Function	***MeanIGD***	***SPREAD***	***CPU Time***	**MeanIGD**	***SPREAD***	***CPU Time***	***MeanIGD***	***SPREAD***	***CPU Time***	***(MeanIGD, SPREAD, CPU Time)***
UF1	0.068187	0.697345	1008.341	0.064189	0.71734	**61.984**	**0.057391**	**0.536142**	341.922	(+, +, -)
UF2	0.024311	0.540971	1014.821	0.026720	0.59107	**32.754**	**0.011047**	**0.428002**	377.031	(+, +, -)
UF3	0.135773	0.666309	1028.630	0.031159	**0.58190**	**53.016**	**0.031120**	0.588970	362.673	(+, -, -)
UF4	0.019737	0.682081	1002.473	0.028890	0.76925	**34.890**	**0.016262**	**0.631167**	310.475	(+, +, -)
UF5	**0.211415**	**0.658579**	1068.459	0.978930	0.72905	**168.892**	0.819614	0.697120	390.539	(-, -, -)
UF6	**0.152526**	**0.694642**	1015.378	0.370197	0.74035	**54.782**	0.311660	0.695705	392.531	(-, -, -)
UF7	0.031356	0.606167	1013.489	0.019931	0.56061	**53.407**	**0.010428**	**0.531020**	320.569	(+, +, -)
UF8	0.536727	0.616086	1104.672	1.836091	0.83692	**60.062**	**0.322575**	**0.583246**	358.638	(+, +, -)
UF9	0.375251	0.703306	1100.592	0.988287	0.71489	**76.625**	**0.108460**	**0.604505**	360.521	(+, +, -)

TABLE 7.2 STATISTICAL SUM OF PROBLEMS FOR WHICH EACH STRATEGY OBTAINS SIGNIFICANTLY BETTER RESULTS

Metric	**NSGA-II-MPX**	**RSMOGA**	**SNOVMOGA**
MeanIGD	02	00	07
SPREAD	02	01	06
CPU Time	00	09	00
Total	04	10	13

Next the focus of the study is moved to decomposition based approach for solving MOPs. Two methods of scalar fitness assignment for decomposition approach Tchebycheffs and Weight-sum are studied. These methods require uniform well distributed weight vectors. In chapter 4 a unique procedure for uniform well distributed weight vector generation using opposition based learning has been presented. A Decomposition based Multi-objective Genetic Algorithm with opposition based learning (DMOGA-OBL) is implemented. Table 7.3 shows performance comparison of DMOGA-OBL with RSMOGA and SNOVMOGA. DMOGA-OBL when compared with RSMOGA and SNOVMOGA has performed better or comparable on UF1-UF7 but not shown improvement in UF8 & UF9. It has good improvement on two functions (UF5 & UF6) having discontinuous Pareto front. SNOVMOGA has shown good SPREAD on six functions UF1, UF4, UF5 & UF7-UF9. RSMOGA is computationally efficient than the other two algorithms. Table 7.4 shows statistics of performance of the RSMOGA, SNOVMOGA and DMOGA-OBL.

The attempts to enhance performance of MOGA with multiparent recombination operators, rank based selection mechanism and decomposition based approaches got remarkably good success in solving highly complex two and three objective MOPs having convex, non-convex and continuous Pareto fronts. The failure, however, with functions UF5 and UF6 having discontinuous Pareto fronts and very complicated Pareto sets is due to the inability of the algorithm to deal with multimodal functions.

In chapter 5 a new study is presented to deal with issues of convergence and diversity in functions having complicated PF and PS. In this study GA procedure has been made more close to natural evolutionary process which incorporate Birth-Reproduction-death cycle. This work has given insight into dying of solutions and its impact on diversity of population. Dying is implicit part of selection process i.e. solutions which are not selected die out. We have proposed and implemented strategies for explicit dying of solution in specific generations. Three strategies for dying of solution from next generation population have been proposed and implemented along with new MOGA framework.

TABLE 7.3 COMPARISON OF DMOGA-OBL WITH OTHER ALGORITHMS

Scheme	**RSMOGA**			**SNOVMOGA**			**DMOGA-OBL**			**Sign test**
Function	***MeanIGD***	***SPREAD***	***CPU Time***	**MeanIGD**	***SPREAD***	***CPU Time***	***MeanIGD***	***SPREAD***	***CPU Time***	***(MeanIGD, SPREAD, CPU Time)***
UF1	0.064189	0.71734	**61.984**	0.057391	**0.536142**	341.922	**0.00991**	0.54605	484.256	(+, -, -)
UF2	0.026720	0.59107	**32.754**	0.011047	0.428002	377.031	**0.00676**	**0.40673**	498.168	(+, +, -)
UF3	0.031159	**0.58190**	**53.016**	**0.031120**	0.588970	362.673	0.03127	0.58931	469.447	(-, -, -)
UF4	0.028890	0.76925	**34.890**	**0.016262**	**0.631167**	310.475	0.01626	0.63560	430.421	(-, -, -)
UF5	0.978930	0.72905	**168.892**	0.819614	**0.697120**	390.539	**0.29061**	0.79242	527.948	(+, -, -)
UF6	0.370197	0.74035	**54.782**	0.311660	0.695705	392.531	**0.03116**	**0.67722**	510.481	(+, +, -)
UF7	0.019931	0.56061	**53.407**	0.010428	**0.531020**	320.569	**0.00481**	0.53516	406.953	(+, -, -)
UF8	1.836091	0.83692	**60.062**	**0.322575**	**0.583246**	358.638	0.836727	0.81782	583.827	(-, -, -)
UF9	0.988287	0.71489	**76.625**	**0.108460**	**0.604505**	360.521	0.891271	0.70783	609.460	(-, -, -)

TABLE 7.4 STATISTICAL SUM OF PROBLEMS FOR WHICH EACH STRATEGY OBTAINS SIGNIFICANTLY BETTER RESULTS

Metric	**RSMOGA**	**SNOVMOGA**	**DMOGA-OBL**
MeanIGD	00	04	05
SPREAD	01	06	02
CPU Time	09	00	00
Total	10	10	07

It is observed that for all three strategies best results for MeanIGD and SPREAD have been obtained for dying rate at or below 50%. Hence we can say that excessive dying of solutions affects the convergence to and diversity in Pareto front. Table 7.5 shows best performance (MeanIGD and SPREAD) of three strategies of dying of solution on function UF1-UF9. From Table 7.5 it is observed that dying strategy ParRem1 with dying rate 50% has outperformed ParRem2 and ParRem3 in terms of SPREAD on four functions (UF2, UF4, UF7 and UF8) out of nine functions. ParRem2 has outperformed ParRem1 and ParRem3 on three functions (UF1 & UF3 with dying rate 50% and UF9 with dying rate 40%). ParRem3 has outperformed ParRem1 and ParRem2 on two functions (UF5 and UF6 with dying rate 50%).

TABLE 7.5 COMPARISON OF PERFORMANCE OF PARREM1, PARREM2 AND PARREM3 ON FUNCTION UF1-UF9 WITH POPULATION SIZE = 100 AND NUMBER OF GENERATIONS = 3000

Scheme	**ParRem1**		**ParRem2**		**ParRem3**		***Sign test***
Function	***MeanIGD (Dying rate)***	***SPREAD (Dying rate)***	***MeanIGD (Dying rate)***	***SPREAD (Dying rate)***	***MeanIGD (Dying rate)***	***SPREAD (Dying rate)***	***(IGD, SPREAD)***
UF1	0.061657 (50%)	0.58012 (50%)	**0.060779 10%)**	**0.55635 (50%)**	0.067689 (10%)	0.55652 (50%)	(- , -)
UF2	**0.019220 (50%)**	**0.40815 (50%)**	0.023877 (50%)	0.45033 (50%)	0.022364 (10%)	0.42707 (30%)	(+ , +)
UF3	0.201176 (10%)	0.52668 (10%)	**0.170225 10%)**	**0.47090 (50%)**	0.180125 (10%)	0.50012 (30%)	(- , -)
UF4	**0.050215 (10%)**	**0.50465 (50%)**	0.054376 (10%)	0.53480 (50%)	0.054034 (10%)	0.55244 (30%)	(+ , +)
UF5	**0.153826 (10%)**	0.67871 (50%)	0.208118 (10%)	0.78428 (50%)	0.201256 (10%)	**0.63878 (50%)**	(+ , -)
UF6	0.064901 (10%)	0.70065 (50%)	0.070440 (10%)	0.76884 (50%)	**0.059221 (30%)**	**0.67963 (50%)**	(- , -)
UF7	**0.083477 (10%)**	**0.67037 (50%)**	0.113827 (30%)	0.79031 (50%)	0.110274 (10%)	0.70431 (50%)	(+ , +)
UF8	0.755344 (50%)	**0.46400 (50%)**	0.738952 (50%)	0.68068 (50%)	**0.738637 (30%)**	0.65029 (10%)	(- , +)
UF9	**0.899907 (30%)**	0.63089 (50%)	0.904564 (10%)	**0.58913 (40%)**	0.959632 (30%)	0.68203 (10%)	(+ , -)

TABLE 7.6 STATISTICAL SUM OF PROBLEMS FOR WHICH EACH STRATEGY OBTAINS SIGNIFICANTLY BETTER RESULTS

Metric	ParRem1	ParRem2	ParRem3
MeanIGD	05	02	02
SPREAD	04	03	02
Total	09	05	04

Table 7.6 shows Statistical Sum of Problems for which each strategy obtains significantly better results. ParRem1 has given statistically significant performance than ParRem2 and ParRem3 and hence dying strategy ParRem1 and dying rate 50% will be used for further experimentation.

The performance of proposed MOGA with dying strategy ParRem1 (MOGA-ParRem1) is compared with proposed MOGA without dying (MOGA-WD) and NSGA-II-MPX. NSGA-II is coded in MatLab 7.1 and in place of SBX operator MPX crossover operator is used. Also parameter setting used for MOGA-WD and NSGA-II-MPX is same as given in section 4.Table 6 shows MeanIGD and SPREAD of MOGA-WD, MOGA-ParRem1 and NSGA-II-MPX. SPREAD values reported in the table 7.7 indicate that MOGA-ParRem1 has outperformed MOGA-WD on functions UF1- UF5 and UF7-UF9. Giving better SPREAD on functions UF1-UF4 and UF8-UF9, MOGA-ParRem1 has outperformed NSGA-II-MPX. IGD measure taken for MOGA-ParRem1 has shown that it has given good performance on functions UF1, UF2 and UF6 functions. NSGA-II-MPX has shown good IGD values on functions UF3-UF5 and UF7-UF9.Table 7.8 shows Statistical Sum of Problems for which each algorithm obtains significantly better results.

TABLE 7.7 PERFORMANCE COMPARISON OF MOGA-ParRem1 WITH MOGA-WD AND NSGA-II-MPX

Scheme	**MOGA-WD**		**MOGA-ParRem1 (DR = 50%)**		**NSGA-II-MPX**		**Sign test**
Functon	**MeanIGD**	**SPREAD**	**MeanIGD**	**SPREAD**	**MeanIGD**	**SPREAD**	**(MeanIGD SPREAD)**
UF1	0.061892	0.615871	**0.061657**	**0.580122**	0.068187	0.697345	(+ , +)
UF2	0.057764	0.644613	**0.019220**	**0.408153**	0.024311	0.540971	(+ , +)
UF3	0.245993	0.583918	0.240011	**0.547451**	**0.135773**	0.666309	(- , +)
UF4	0.066059	0.550369	0.062340	**0.504653**	**0.019737**	0.682081	(- , +)
UF5	1.046338	0.753780	0.214201	0.678715	**0.211415**	**0.658579**	(- , -)
UF6	0.230648	0.695138	**0.072117**	0.700656	0.152526	**0.694642**	(+, -)
UF7	0.101259	0.715222	0.098397	0.670374	**0.031356**	**0.626167**	(- , -)
UF8	0.836721	0.686384	0.755344	**0.464006**	**0.536727**	0.616086	(- , +)
UF9	0.975251	0.883306	0.914492	**0.630897**	**0.375251**	0.703306	(- , +)

TABLE 7.8 STATISTICAL SUM OF PROBLEMS FOR WHICH EACH STRATEGY OBTAINS SIGNIFICANTLY BETTER RESULTS

Metric	**MOGA-WD**	**MOGA-ParRem1 (dying rate = 50%)**	**NSGA-II-MPX**
MeanIGD	0	03	06
SPREAD	0	06	03
Total	0	09	09

TABLE 7.9 COMPARISON OF MOGA-PARREM1 WITH OTHER ALGORITHMS

Scheme	**RSMOGA**			**SNOVMOGA**			**DMOGA-OBL**			**MOGA-ParRem1 (DR = 50%)**			**Sign test**
Function	***MeanIGD***	***SPREAD***	***CPU Time***	**MeanIGD**	***SPREAD***	***CPU Time***	***MeanIGD***	***SPREAD***	***CPU Time***	***MeanIGD***	***SPREAD***	***CPU Time***	***(MeanIGD, SPREAD, CPU Time)***
UF1	0.064189	0.71734	**61.98**	0.057391	**0.536142**	341.92	**0.00991**	0.54605	484.25	0.061657	0.58012	169.23	(-,-,-)
UF2	0.026720	0.59107	**32.75**	0.011047	0.428002	377.03	**0.00676**	**0.40673**	498.16	0.019220	0.40815	192.11	(-, -, -)
UF3	0.031159	0.58190	**53.01**	**0.031120**	0.588970	362.67	0.03127	0.58931	469.44	0.201176	**0.52668**	198.42	(-,+,-)
UF4	0.028890	0.76925	**34.89**	**0.016262**	0.631167	310.47	0.01626	0.63560	430.42	0.050215	**0.50465**	174.54	(-,+,-)
UF5	0.978930	0.72905	**168.8**	0.819614	0.697120	390.53	0.29061	0.79242	527.94	**0.153826**	**0.67871**	205.71	(+,+,-)
UF6	0.370197	0.74035	**54.78**	0.311660	0.695705	392.53	**0.03116**	**0.67722**	510.48	0.064901	0.70065	186.34	(-, -, -)
UF7	0.019931	0.56061	**53.40**	0.010428	**0.531020**	320.56	**0.00481**	0.53516	406.95	0.083477	0.67037	197.19	(-, -, -)
UF8	1.836091	0.83692	**60.06**	**0.322575**	0.583246	358.63	0.836727	0.81782	583.82	0.755344	**0.46400**	220.16	(-, +,-)
UF9	0.988287	0.71489	**76.62**	**0.108460**	**0.604505**	360.52	0.891271	0.70783	609.46	0.899907	0.63089	234.03	(-, -, -)

TABLE 7.10 STATISTICAL SUM OF PROBLEMS FOR WHICH EACH STRATEGY OBTAINS SIGNIFICANTLY BETTER RESULTS

Metric	**RSMOGA**	**SNOVMOGA**	**DMOGA-OBL**	**MOGA-ParRem1 (dying rate = 50%)**
MeanIGD	00	04	04	01
SPREAD	00	03	02	04
CPU Time	09	00	00	00
Total	09	07	06	05

Empirically it is found that Among the three strategies of dying, strategy ParRem1 has given statistically significant results on 4 functions, ParRem2 and ParRem3 have given statistically significant performance on three & two functions respectively in terms of SPREAD metric and hence it is concluded that no single strategy is capable enough to solve all the functions (UF1-UF9) properly. In chapter 6 an ensemble of dying strategy based MOGA is presented to improve performance of MOGA. Ensemble learning is a machine learning paradigm and has proven to be very efficient and effective for adjusting algorithmic control parameters and operators in an online manner. EDS-MOGA maintains a pool of three dying strategies and appropriate strategy is selected for solving a problem.

Performance Comparison of EDS-MOGA with ParRem1, ParRem2 and ParRem3 is shown in Table 7.11. EDS-MOGA has outperformed MOGA with single dying strategy i.e.ParRem1, ParRem2 and ParRem3 on functions UF1-UF5 & UF7-UF9 in terms of IGD metric and on functions UF2, UF5-UF9 functions in terms of SPREAD metric. EDS-MOGA combines' features of three dying strategy and contribution of each strategy has produced good convergence and diversity in the population.

TABLE 7.11 PERFORMANCE COMPARISON OF EDS-MOGA WITH ParRem1, ParRem 2 AND ParRem 3

Scheme	**ParRem1**		**ParRem2**		**ParRem3**		**EDS-MOGA**	
Function	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***	***MeanIGD***	***SPREAD***
UF1	0.061657	0.58012	0.060779	0.55635	0.067689	0.55652	**0.054910**	0.55891
UF2	0.019220	0.40815	0.023877	0.45033	0.022364	0.42707	**0.013663**	**0.40647**
UF3	0.201176	0.52668	0.170225	0.47090	0.180125	0.50012	**0.097539**	0.49136
UF4	0.050215	0.50465	0.054376	0.53480	0.054034	0.55244	**0.016117**	0.50529
UF5	0.153826	0.67871	0.208118	0.78428	0.201256	0.63878	**0.094909**	**0.53358**
UF6	0.064901	0.70065	0.070440	0.76884	0.059221	0.67963	0.060966	**0.66912**
UF7	0.083477	0.67037	0.113827	0.79031	0.110274	0.70431	**0.080293**	**0.66937**
UF8	0.755344	0.46400	0.738952	0.68068	0.738637	0.65029	**0.271814**	**0.46076**
UF9	0.899907	0.63089	0.904564	0.58913	0.959632	0.68203	**0.165976**	**0.55382**

Performance of EDS-MOGA is compared with NSGA-II-MPX, RSMOGA, SNOVMOGA, DMOGA-OBL and MOGA-ParRem1. Table 7.12, table 7.13 and table

7.14 compares performance of EDS-MOGA with other algorithms in terms of MeanIGD, SPREAD and CPU Time metrics respectively. Sign test is also performed and shown in last column. A + indicate good or comparable performance and – indicate insignificant performance. EDS-MOGA has shown remarkably good performance on UF5 function in terms of MeanIGD and SPREAD metric. UF5 has a discrete and uniformly distributed Pareto front and it is hard for an algorithm to deal with. EDS-MOGA has successfully moved few solutions on the Pareto front and remaining solutions are very close to Pareto front. EDS-MOGA has given good or comparable spread on all functions except UF7. Thus EDS-MOGA has given improved performance than MOGA-ParRem1 (with single dying strategy). This is because all the three dying strategies contributed in maintaining convergence and diversity in population in the initial generation and has helped in proper exploration of search space. The algorithm selects one suitable dying strategy to evolve solutions in later generations. Hence better convergence and diversity is seen in the end.

TABLE 7.12 PERFORMANCE COMPARISON OF EDS-MOGA WITH THE OTHER ALGORITHMS

Scheme	**RSMOGA**	**SNOVMOGA**	**DMOGA-OBL**	**MOGA-ParRem1**	**EDS-MOGA**	**NSGA-II-MPX**	**Sign test**
Function	**MeanIGD**	***MeanIGD***	***MeanIGD***	***MeanIGD***	***MeanIGD***	***MeanIGD***	
UF1	0.064189	0.057391	**0.00991**	0.061657	0.054910	0.068187	(-)
UF2	0.026720	0.011047	**0.00676**	0.019220	0.010663	0.024311	(-)
UF3	0.031159	**0.031120**	0.03127	0.201176	0.097539	0.135773	(-)
UF4	0.028890	0.016262	0.01626	0.050215	**0.016117**	0.019737	(+)
UF5	0.978930	0.819614	0.29061	0.153826	**0.094909**	0.211415	(+)
UF6	0.370197	0.311660	0.03116	0.064901	**0.029566**	0.152526	(+)
UF7	0.019931	0.010428	**0.00481**	0.083477	0.080293	0.031356	(-)
UF8	1.836091	0.322575	0.836727	0.755344	**0.271814**	0.536727	(+)
UF9	0.988287	**0.108460**	0.891271	0.899907	0.125976	0.375251	(+)

TABLE 7.13 PERFORMANCE COMPARISON OF EDS-MOGA WITH THE OTHER ALGORITHMS

Scheme	**RSMOGA**	**SNOVMOGA**	**DMOGA-OBL**	**MOGA-ParRem1**	**EDS-MOGA**	**NSGA-II-MPX**	**Sign test**
Function	***SPREAD***	***SPREAD***	***SPREAD***	***SPREAD***	***SPREAD***	***SPREAD***	
UF1	0.71734	**0.536142**	0.54605	0.58012	0.54891	0.697345	(+)
UF2	0.59107	0.428002	0.40673	0.40815	**0.40647**	0.540971	(+)
UF3	0.58190	0.588970	0.58931	0.52668	**0.52136**	0.666309	(+)
UF4	0.76925	0.631167	0.63560	0.50465	**0.50429**	0.682081	(+)
UF5	0.72905	0.697120	0.79242	0.67871	**0.53358**	0.658579	(+)
UF6	0.74035	0.695705	0.67722	0.70065	**0.66912**	0.694642	(+)
UF7	0.56061	**0.53102**	0.53516	0.67037	0.65037	0.606167	(-)
UF8	0.83692	0.583246	0.81782	0.46400	**0.46076**	0.616086	(+)
UF9	0.71489	0.604505	0.70783	0.63089	**0.55382**	0.703306	(+)

TABLE 7.14 PERFORMANCE COMPARISON OF EDS-MOGA WITH THE OTHER ALGORITHMS

Scheme	**RSMOGA**	**SNOVMOGA**	**DMOGA-OBL**	**MOGA-ParRem1**	**EDS-MOGA**	**NSGA-II-MPX**	**Sign test**
Function	***CPU Time***	***CPU Time***	***CPU Time***	***CPU Time***	***CPU Time***	***CPU Time***	
UF1	**61.984**	341.922	484.256	169.232	178.167	1008.341	(-)
UF2	**32.754**	377.031	498.168	192.114	198.810	1014.821	(-)
UF3	**53.016**	362.673	469.447	198.426	198.602	1028.630	(-)
UF4	**34.890**	310.475	430.421	174.547	185.118	1002.473	(-)
UF5	**168.892**	390.539	527.948	205.712	234.311	1068.459	(-)
UF6	**54.782**	392.531	510.481	186.348	197.801	1015.378	(-)
UF7	**53.407**	320.569	406.953	197.195	201.387	1013.489	(-)
UF8	**60.062**	358.638	583.827	220.169	241.617	1104.672	(-)
UF9	**76.625**	360.521	609.460	234.036	251.104	1100.592	(-)

The major contribution of this study is the introduction of Dying strategy for removal of solutions for formation of next generation. Dying has been made explicit part of new MOGA framework and a unique ensemble of dying strategies for performance improvement of MOGA has been proposed.

7.2 Conclusions

The test problems simulate the various difficulty levels of real-world optimization problems. The study has attempted to device ways to enhance the performance MOGAs in solving such problems of different characteristics very successfully. To this end, the work has introduced multi-parent recombination operators and rank based selection schemes and also proposed new MOGAs to investigate their behaviors. Also decomposition based MOGA with opposition based learning has been proposed to address the issue of convergence and diversity. The crux of the research work is development of new strategies and improvement of MOGAs with a view to solve, highly complex MOPs having various difficulties of real world, in an efficient manner. The RSMOGA is the most efficient MOGA with lowest CPU Time it has taken to solve all the problems as shown in table 7.14. SNOVMOGA has shown good performance on functions having continues PFs but failed on functions having discontinues PFs. It has given remarkably good SPREAD on UF1 & UF7 function. DMOGA-OBL has shown better convergence on functions UF1, UF2 & UF7 as shown in table 7.13. Finally unique schemes of dying of solutions are designed to curb mass dying of solutions in early generations thereby reducing diversity and hindering convergence. Ensemble of three dying strategies has shown improved performance on 2 & 3-objective MOPs having convex, concave, continues and discontinues PFs and comes out to be champion of all with respect to SPREAD metric as evident from table 7.13.

Finally, we conclude that the in this work successful attempts have been made to improve performance of MOGA for solving MOPs showing difficulty of real world problems. Such study will definitely be helpful for one to choose and modify an algorithm for solving their problems. The experimental results also suggest that the presence of many local Pareto optimal solutions in MOPs with complicated PSs could be very challenging for algorithms.

The scope for future research:

- The strengths and weaknesses of these algorithms should be thoroughly studied on test problems with different characteristics.
- Checking the effect of decision space dimensionality on the optimization process (scale up study)
- Multi-objective Genetic Algorithms designed and implemented in this work can be used to solve Many objective optimization problems i.e. MOPs having 5 or more objectives and constrained optimization problems
- Thorough experimentation with other evolutionary computation techniques such as particle swarm or ant colony for solving multi-objective optimization with many competing objectives is an interesting future work direction
- The topic that deserves further research is fine tuning proposed dying strategies and design of new dying strategies
- Ensemble learning based MOGA can be extended by incorporating multiple ensembles of operators like crossover or mutation

References

1. Deb K.,“Multi-Objective Optimization using Evolutionary Algorithms”, Chichester: John Wiley & Sons ISBN 0-471-87339-X. 2001

2. Hillier, F. S. and Lieberman, G. J., “Introductions to Operations Research”, 7th edition, McGraw-Hill 2001

3. F. Y. Edgeworth, “*Mathematical Physics”,* P. Keagan, London, England, 1881.

4. I. Das and J. Dennis., “*Normal-boundary intersection*: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems”, *SIAM Journal of Optimization*, 8:631–657, 1998

5. I. Das and J. E. Dennis, “A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems”, *Structural and Multidisciplinary Optimization*, 14(1):63–69, August, 1997

6. Michael Dellnitz, Oliver Sch¨utze, and T. Hestermeyer, “Covering Pareto Sets by Multilevel Subdivision Techniques”, *Journal of optimization theory and applications*, 124(1):113–136, 2005

7. Y.Y. Haimes, L.S. Lasdon, and D.A. Wismer, “On a bicriterion formulation of the problems of integrated system identification and system optimization”, *IEEE ransactions on Systems, Man, and Cybernetics*, 1(3):296–297, 1971

8. Kaisa M Miettinen, “ *Nonlinear Multiobjective Optimization”,* Springer, 1999.

9. Ulrich Bodenhofer, , “Genetic Algorithms: Theory and Applications”, Lecture Notes Third Edition 2003/2004

10. Q. Zhang, Aimin Zhou, S. Z. Zhao, P. N. Suganthan, Wudong Liu, and S. Tiwari, “Multi-objective optimization Test Instances for the CEC 2009 Special Session and Competition,” University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report 2008.

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

12. Zitzler E., K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: Empirical results,” *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.

13. F. Kursawe, “A variant of evolution strategies for vector optimization,”in *Parallel Problem Solving from Nature*, H.-P. Schwefel and R. Männer, Eds. Berlin, Germany: Springer-Verlag, pp. 193–197. 1990,

14. B¨ack, T., Fogel, D., & Michalewicz, Z., “Handbook of Evolutionary Computation”, Institute of Physics Publishing Ltd, Bristol and Oxford University Press, New York, 1997.

15. Goldberg D.E., “Genetic Algorithms in Search, Optimization and Machine Learning”, Pearson Education Asia, 1989.

16. Carlos Fonseca.1995, “Multiobjective Genetic Algorithms with Application to Control Engineering Problems”, PhD thesis, Department of Automatic Control and System Engineering The University of Sheffield,1995.

17. Deb, K., A. Pratap, S. Agarwal, T. Meyarivan, “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II”, IEEE Transactions on Evolutionary Computation, Vol. **6**, No 2, 182-197. Apr. 2002

18. Zitzler, E. and Thiele, L.:, Multiobjective optimization using evolutionary algorithms: a comparative study, in A. E. Eiben (ed.), Proceedings of the Parallel Problem Solving from Nature V Conference, Springer-Verlag, Berlin, pp. 292–301,1998

19. Q. Zhang and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731. 2007

20. William Hart, Natalio Krasnogos, and Jim Smith. 2005, Recent Advances in Memetic Algorithms, volume 166 of Studies in Fuzziness and Soft Computing. Springer, 2005.

21. Herrara, F., Lozano, M. and Verdegay, J. L, Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis, *Artificial Intelligence Review* 12, 265 – 319, 1998

22. Eshelman L.J. and Schaffer J.D., ”Real-coded genetic algorithms and interval schemata”, In D. Whitley (Ed.), Foundation of Genetic Algorithm II, 187-202.evolution strategies, in E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello and D. Corne(eds), Proceedings of the First

International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), Springer-Verlag, Berlin, pp. 96–110. 1993,

23. Deb K., Agrawal R.B., “Simulated binary crossover for continuous search space”, Complex System 9 115-148. 1995,

24. M.M. Raghuwanshi, O.G. Kakde, P.M. Singru, U. Kale, “Simulated Binary Crossover with Lognormal Distribution”, In Proceedings of the 7th Asia-Pacific Conference on Complex Systems (Complex 2004) 6-10 Dec. 2004.

25. Eiben, A.E., “Multi-parent recombination”, In T. Bäck, D.B. Fogel, and Z. Michalewicz, editors, Evolutionary Computation 1: Basic Algorithms and Operators, pages 289-307, Institute of Physics Publishing, 2000,

26. Ono I. & Kobayashi S., “A real-coded genetic algorithm for functional optimization using unimodal normal distribution crossover”, In Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA-7) 246-253, 1997.

27. Tsutsui, S. and Ghosh, A.,“A study on effect of Multi-parent recombination in real-coded genetic algorithm”, proceedings of the 1998 IEEE, CEC, pp. 828-833, 1998.

28. Tsutsui, S., Yamamura, M. and Higuchi, T., “Multi-parent recombination with simplex crossover in real-coded genetic algorithms”, In: Proc. of the Genetic and Evolutionary Computation Conference(GECCO-99), Morgan Kaufmann, San Mateo, CA, pp. 657–664, 1999.

29. Kita, H., Ono I. and Kobayashi, S., “Multi-parent extension of the Unimodal Normal Distribution Crossover for real-coded genetic algorithm”, In proc. of the International conference on evolutionary computation’99 (IEEE press, Piscataway, New Jersey, 1999), pp. 646-651, 1999

30. Deb, K., Anand, A. and Joshi, D.,“A computationally efficient evolutionary algorithm for real parameter optimization”, Evolutionary Computation Journal 10(4): 371-395. 2002,

31. M.M. Raghuwanshi and O. G. Kakde, “Multi-parent Recombination operator with Polynomial or Lognormal Distribution for Real Coded Genetic Algorithm” *2nd Indian International Conference on Artificial Intelligence (IICAI),* pp. 3274-3290, 2005.

32. Juan Durillo, Antonio Nebro, Francisco Luna, and Enrique Alba., “On the effect of the steady-state selection scheme in multi-objective genetic algorithms”, In Proceedings of

the 2009 International Conferenceo n Evolutionary Multi-criterion Optimization, volume 5467 of Lecture Notes in Computer Science, page 183197. Springer, 2009

33. Marco Laumanns, Eckart Zitzler, and Lothar Thiele., "On the effects of archiving, elitism, and density based selection in evolutionary multiobjective optimization", In The 1st International Conference on Evolutionary Multi-Criterion Optimization, volume 1993 of Lecture Notes in Computer Science. Springer, 2001.
34. Deb, K. and Goldberg, D. E, "An investigation of niche and species formation in genetic function optimization", in J. D. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 42 – 50.1989
35. Schaffer, J. D, "Multiple objective optimization with vector evaluated genetic algorithms", in J. Grefenstette (ed.), Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Mahwah, New Jersey, pp. 93–100,1985
36. Fourman, M. P, "Compaction of symbolic layout using genetic algorithms", in J. Grefenstette (ed.), Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Mahwah, New Jersey, pp. 141–153, 1985
37. Fleming, P. J. and Pashkevich, "A. P, Computer aided control system design using a multiobjective optimization approach", Proceedings of the IEE Control '85 Conference, pp. 174–179.1985
38. Haleja, P. and Lin, C.-Y., "Genetic search strategies in multicriterion optimal design", Structural Optimization 4, 99–107. 1992
39. Jin, Y., Okabe, T. and Sendhoff, B., "Adapting weighted aggregation for multiobjective evolutionary optimization", Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001) San Francisco, California, pp. 1042–1049. 2001
40. Jin, Y., Okabe, T. and Sendhoff, B., "Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how?", in L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon and E. Burke (eds), Proceedings of the 2001 Genetic and Evolutionary Computation Conference (GECCO 2001), Morgan Kaufmann Publishers, San Francisco, California, pp. 1042–1049.2001

41. Coello, C. A. C., “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art”, Computer Methods in Applied Mechanics and Engineering 191(11–12), 1245–1287. 2002

42. Fonseca, C. M. and Fleming, P. J., “Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization”, in S. Forrest (ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kauffman Publishers, San Mateo, California, pp. 416–423. 1993

43. Horn, J. and Nafpliotis, N, “Multiobjective optimization using the niched Pareto genetic algorithm”, IlliGAL Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois. 1993

44. Srinivas, N., K. Deb, “ Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms’, Evolutionary Computation, Vol. **2**, No 3, 221-248, 1994.

45. Fonseca, C. M. and Fleming, P. J., Multiobjective genetic algorithms made easy: Selection, sharing, and mating restriction, in A. M. S. Zalzala (ed.), Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA 95), Institution of Electrical Engineers, Stevenage, UK, pp. 42–52. 1995

46. Goldberg, D. E. and Richardson, J., “Genetic algorithms with sharing for multimodal function optimization”, in J. Grefenstette (ed.), Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Mahwah, New Jersey, pp. 41–49. 1987

47. Knowles and D. Corne, "The Pareto archived evolution strategy: a new baseline algorithm for Pareto multi-objective optimisation," in Proceedings of the Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, pp. 98-105. , 1999

48. Zitzler, E., Laumanns, M. and Bleuler, S., “A tutorial on evolutionary multiobjective optimization”, in X. Gandibleux, M. Sevaux, K. S¨orensen and V. T’Kindt (eds), *Methaheuristics for Multiobjective Optimisation*, Vol. 535 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin.2004

49. Rudolph, G. and Agapie, A., “Convergence properties of some multi-objective evolutionary algorithms”, *Proceedings of the Congress on Evolutionary Computation (CEC) 2000*, IEEE, IEEE Press, pp. 1010 – 1016.2000

50. Abbass, H. A., Sarker, R. and Newton, C., “ PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems, in IEEE Neural Networks Council (ed.), *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*, Vol. 2, IEEE Service Center, Piscataway, New Jersey, pp. 971–978. and R. M¨anner (eds), Proceedings of the Parallel Problem Solving from Nature I Conference, Springer-Verlag, Berlin, pp. 193–197.2001

51. Sarker, R., Liang, K. and Newton, C., “A new evolutionary algorithm for multiobjective optimization”, *European Journal of Operational Research* 140(1), 12–23. 2002

52. Deb, K., Mohan, M. and Mishra, S., “Towards a quick computation of well-spread Pareto-optimal solutions”, in C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb and L.Thiele (eds), *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, Springer-Verlag, Berlin, pp. 222–236.2003

53. Laumanns, M., Thiele, L., Deb, K. and Zitzler, E., “Combining convergence and diversity in evolutionary multi-objective optimization”, *Evolutionary Computation* 10(3),263–282. 2002

54. Papadimitriou, C. H. and Yannakakis, M.:, “On the approximability of trade-offs and optimal access of Web sources, *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2000)*, IEEE Computer Society, Los Alamitos, California, pp. 86–92. 2000

55. Bosman, P. A. N. and Thierens, D.:, “The balance between proximity and diversity in multiobjective evolutionary algorithms”, *IEEE Transactions on Evolutionary Computation* 7(2), 174–188. 2003

56. Deb, K. and Goel, T.:, “Controlled elitist non-dominated sorting genetic algorithms for better convergence”, in E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello and D. Corne (eds), *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, Springer-Verlag, pp. 67–81. 2001

57. Laumanns, M., Zitzler, E. and Thiele, L., “On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization”, in E. Zitzler, K.

Deb, L. Thiele, C. A. C. Coello and D. Corne (eds), Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), Springer-Verlag, Berlin, pp. 181–196. 2001

58. Laumanns, M. and Ocenasek, J.:, "Bayesian optimization algorithms for multi objective optimization, in J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacanas and H.-P. Schwefel (eds*), Proceedings of the Parallel Problem Solving from Nature VII Conference*, Springer-Verlag, Berlin, pp. 298–307. 2002

59. Wienke, D., Lucasius, C. and Kateman, G.:, Multicriteria target vector optimization of analytical procedures using a genetic algorithm - part I. theory, numerical simulations and application to atomic emission spectroscopy, *Analytica Chimica Acta* 265, 211 – 225. 1992

60. Lohn, J. D., Kraus, W. F. and Haith, G. L.:, Comparing a coevolutionary genetic algorithm for multiobjective optimization, in IEEE Neural Networks Council (ed.), *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, Vol. 2, IEEE Service Center, Piscataway, New Jersey, pp. 1157–1162. 2002

61. Zitzler, E., M. Laumanns, L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm", In: EUROGEN 2001.

62. Zitzler E. and Künzli S., "Indicator-Based Selection in Multiobjective Search. In X. Yao et al., editors, *Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *LNCS*, pages 832–842. Springer, 2004

63. D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope-based selection algorithm for multi-objective optimization," in Parallel Problem Solving from Nature PPSNVI. 6th International Conference. Proceedings (Lecture Notes in Computer Science Vol.1917), Paris, France, pp. 839-48. , 2000

64. Bentley, P.J. and Wakefield, J.P.' "Finding acceptable solutions in the Pareto-optimal range using multi-objective geneticalgorithms", In (Chawdry et al, eds.) Soft Computing in Eng'g Design and Manufacturing, Springer Verlag, 1997.

65. Drechsler, D., Drechsler, R., Becker, B. "Multi-objective optimisation based on relation favour". In Proc. 1st EMO, pp.154–166, Springer Ver-lag, 2001.

66. di Pierro, K. Soon-Thiam, and D. A. Savic, "An investigation on preference order ranking scheme for multi-objective evolutionary optimization," IEEE Transactions on Evolutionary Computation, vol. 11, pp. 17-45, 2007.

67. di Pierro, F., Djordjevic, S., Khu, S.-T, Savic, D. and Walters, G.A. Automatic calibration of urban drainage model using a novel multi-objective GA. In Krebs & Fuchs (eds.) Urban Drainage Modelling‘04, pp. 41–52, 2004.

68. Maneeratana, K., Boonlong, K. and Chaiyaratana, N.,“Compressed-objective genetic algorithm”, In PPSN IX, SpringerLNCS, pp. 473–482, 2006.

69. J. Knowles and D. Corne, “Techniques for Highly Multiobjective Optimisation: Some Nondominated Points are Better than Others”. GEC-CO‘07, London, England, United Kingdom. July 7–11, 2007,

70. B. Y. Qu, P. N. Suganthan, “Multi-objective Evolutionary Programming without Non-domination Sorting is up to Twenty Times Faster”, 2009 *Congress on Evolutionary Computation*-CEC09, 2009

71. BY. Qu and P. N. Suganthan, “Multi-objective evolutionary algorithms based on the summation of normalized objectives and diversified selection”, Information sciences, vol. 180, pp. 3170-3181. 2010.

72. B.Y. Qu and Suganthan P. N., “Multi-Objective Differential Evolution based on the Summation of Normalized Objectives and Improved Selection Method”, SDE11, 2011

73. Santosh Tiwari, Georges Fadel, Patrick Koch, and Kalyanmoy Deb, “Performance Assessment of the Hybrid Archive-based Micro Genetic Algorithm (AMGA) on the CEC09 Test Problems”, Congress on Evolutionary Computation-CEC09,2009

74. H. R. Tizhoosh, “Opposition-based learning: A new scheme for machine intelligence,” in Proceedings - International Conference on Computational Intelligence for Modelling, Control and Automation, CIMCA and International Conference on Intelligent Agents, Web Technologies and Internet, vol. 1, pp. 695–701, 2005.

75. Fares S. Al-Qunaieer, Hamid R. Tizhoosh, Shahryar Rahnamayan, “Opposition Based Computing – A Survey”, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,978-1-4244-8126-2/10/ ©2010

76. S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, “Opposition-Based Differential Evolution Algorithms”, IEEE Congress on Evolutionary Computation (CEC), Vancouver, Canada, pp. 2010-2017, Sept. 2006.

77. S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, “Quasi-Oppositional Differential Evolution”, IEEE Congress on Evolutionary Computation (CEC), Singapore, pp. 22- 29, Sept. 2007.

78. S. Rahnamayan, G.G. Wang, “Center-Based Sampling for Population-Based Algorithms”, IEEE Congress on Evolutionary Computation (CEC), pp. 933-938, May 2009.

79. L. Peng, Y.Wang, “Differential Evolution using Uniform-Quasi-Opposition for Initializing the Population”, Information Technology Journal, Vol. 9, No. 8, pp. 1629-1634, 2010.

80. L. Peng, Y. Wang, G. Dai, “A Novel Opposition-Based Multi-objective Differential Evolution Algorithm for Multi-objective Optimization”, Advances in Computation and Intelligence, Vol. 5370, pp. 162-170, Springer Berlin and Heidelberg, 2008.

81. L. Han, X. He, “A Novel Opposition-Based Particle Swarm Optimization for Noisy Problems”, Third International Conference on Natural Computation (ICNC), Haikou,Vol. 3, pp. 624-629, Aug. 2007.

82. A.R. Malisia, H.R. Tizhoosh, “Applying Opposition-Based Ideas to the Ant Colony System, IEEE Swarm Intelligence Symposium (SIS), Honolulu, Hawaii, pp. 182-189,Apr. 2007.

83. H. Ishibuchi and T. Murata, “Multiobjective genetic local search algorithm and its application to flowshop scheduling,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, no. 3, pp.392–403, 1998.

84. A. Jaszkiewicz, “On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment,” *IEEE Trans. Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, Aug. 2002.

85. H. Ishibuchi, T. Yoshida, and T. Murata, “Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling,” *IEEE Trans. Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, Apr. 2003.

86. E. J. Hughes, “Multiple single objective pareto sampling,” in *Proc. Of Congress on Evolutionary Computation (CEC’03)*, Canberra, pp. 2678–2684. 2003

87. Maria João Alvesa,, Marla Almeidab, “MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem”, Computers & Operations Research 34 (2007) 3458 -- 3470 2006 Elsevier Ltd. All rights reserved.doi:10.1016/j.cor.2006.02.008, 2007

88. en.wikipedia.org/wiki/Death

89. en.wikipedia.org/wiki/Extinction

90. Rahila Patel, M.M.Raghuwanshi, L.G.Malik “An Improved Ranking Scheme For Selection Of Parents In Multi-Objective Genetic Algorithm”, IEEE International Conference on Computer security and network technology (CSNT) 2011, Shri Mata Vaishnav Devi University, Katra, Jammu (J&K), on pages 734-739, DOI: 10.1109/CSNT.2011.156, June 3-5, 2011

91. Aimin Zhoua, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan ,Qingfu Zhang, "Multiobjective Evolutionary Algorithms: A Survey of the State-of-the-art", Swarm and Evolutionary Computation, Vol. 1, No. 1, pp. 32-49, Mar 2011.

92. David H. Wolpert and William G. Macready, “No Free Lunch Theorems for Optimization”, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 1, NO. 1, APRIL 1997

93. S. Z. Zhao, P. N. Suganthan, Q. Zhang, "Decomposition Based Multiobjective Evolutionary Algorithm with an Ensemble of Neighborhood Sizes", *IEEE Trans. on Evolutionary Computation, Vol. 16, No. 3, pp. 442-446, June 2012.*

94. E. L. Yu and P. N. Suganthan, “Ensemble of niching algorithms,” *Inform.Sci.*, vol. 180, no. 15, pp. 2815–2833, Aug. 2000.

95. R. Mallipeddi and P. N. Suganthan, “Ensemble of constraint handling techniques,” *IEEE Trans. Evol. Computat.*, vol. 14, no. 4, pp. 561–579,Aug. 2010.

96. S. Z. Zhao and P. N. Suganthan, “Multi-objective evolutionary algorithm with ensemble of external archives,” *Int. J. Innovative Comput., Inform. Contr.*, vol. 6, no. 1, pp. 1713–1726, Apr. 2010.

97. Bo-Yang Qu, Ponnuthurai Nagaratnam Suganthan, “Novel Multimodal Problems and Differential Evolution with Ensemble of Restricted Tournament Selection”, in the proc. IEEE confrernce on Congress on Evolutionary Computation CEC2010

98. B. Y. Qu and P. N. Suganthan, “Constrained Multi-Objective Optimization Algorithm with Ensemble of Constraint Handling Methods”, School of Electrical and Electronic Engineering Nanyang Technological University, Singapore Available at http://www.ntu.edu.sg/home/epnsugan/index_files/EEAs-EOAs.htm

99. Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan, “ Differential Evolution Algorithm with Ensemble of Parameters and Mutation and Crossover Strategies”,Nanyang Technological university, Singapore, Available at http://www.ntu.edu.sg/home/epnsugan/index_files/EEAs-EOAs.htm

100. Q. Zhang, W. Liu, and H. Li, “The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances,” in *Proc. CEC*, 2009, pp. 203–208.

101. Gavin Brown, Ensemble Learning Encyclopedia of Machine Learning, C.Sammut & G.I.Webb (Eds.), Springer Press 2010

102. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, Adaptive Mixtures of Local Experts. Neural Computation, 3(1):79-87, 1991.

Appendix

ZDT6	10	[0,1]	$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(x) = g(x)\left[1 - (f_1(x)/g(x))^2\right]$ $g(x) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right]^{0.25}$	$x_1 \in [0,1]$ $x_i = 0,$ $i = 2,\ldots,n$	Nonconvex Non-uniformly spaced

IEEE CEC09 Test Problems		
Problem	**No. of Objective**	**Objectives and PS**
UF1	2	$f_1 = x + \frac{2}{\lvert J_1 \rvert}\sum_{j \in J_1}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2 \rvert}\sum_{j \in J_2}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ $where J_1 = \{j \mid j\ is odd\ and\ 2 \le j \le n\}$ $and\ J_2 = \{j \mid j\ is\ even\ and\ 2 \le j \le n\}$ $Its\ PF\ is \quad f_2 = 1 - \sqrt{f_1}, \quad 0 \le f_1 \le 1$ $PS\ is \quad x_j = \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), \; j = 2,\ldots,n, \quad 0 \le x_1 \le 1$

UF2	2	$f_1 = x_1 + \frac{2}{\lvert J_1 \rvert} \sum_{j \in J_1} y_j^2$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2 \rvert} \sum_{j \in J_2} y_j^2$ $where J_1 = \{ j \mid jisoddand\, 2 \le j \le n \} and J_2 = \{ j \mid jisevenand\, 2 \le j \le n \}$ $y_j = \begin{cases} x_j - \left[0.3 x_1^2 \cos\left(24\pi x_1 + \frac{4 j\pi}{n} \right) + 0.6 x_1 \right] \cos\left(6\pi x_1 + \frac{j\pi}{n} \right) j \in J_1 \\ x_j - \left[0.3 x_1^2 \cos\left(24\pi x_1 + \frac{4 j\pi}{n} \right) + 0.6 x_1 \right] \cos\left(6\pi x \ + \frac{j\pi}{n} \right) j \in J_2 \end{cases}$ *Its PF is* $f_2 = 1 - \sqrt{f_1}, \quad 0 \le f_1 \le 1$ *Its PS is* $x_j = \begin{cases} \left\{ 0.3 x_1^2 \cos\left(24\pi x_1 + \frac{4 j\pi}{n} \right) + 0.6 x_1 \right\} \cos\left(6\pi x_1 + \frac{j\pi}{n} \right) & j \in J_1 \\ \left\{ 0.3 x_1^2 \cos\left(24\pi x_1 + \frac{4 j\pi}{n} \right) + 0.6 x_1 \right\} \sin\left(6\pi x_1 + \frac{j\pi}{n} \right) & j \in J_2 \end{cases}$ $0 \langle x_1 \langle 1$
UF3	2	$f_1 = x_1 + \frac{2}{\lvert J_1 \rvert} \left(4 \sum_{j \in J_1} y_j^2 - 2 \prod_{j \in J_1} \cos\left(\frac{20 y_j \pi}{\sqrt{j}} \right) + 2 \right)$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2 \rvert} \left(4 \sum_{j \in J_2} y_j^2 - 2 \prod_{j \in J_2} \cos\left(\frac{20 y_j \pi}{\sqrt{j}} \right) + 2 \right)$ *where* J_1 *and* J_2 *are the same as those of UF1 and* $y_j = x_j - x_1^{0.5\left(1.0 + \frac{3(j-2)}{n-2}\right)}, \quad j = 2, \ldots., n$ *Its PF is* $f_2 = 1 - \sqrt{f_1}, \quad 0 \le f_1 \le 1$ *Its PS is* $x_j = x_1^{0.5\left(1.0 + \frac{3(j-2)}{n-2}\right)}, \quad j = 2, \ldots., n. \quad 0 \le x_1 \le 1$
UF4	2	$f_1 = x_1 + \frac{2}{\lvert J_1 \rvert} \sum_{j \in J_1} h(y_j)$ $f_2 = 1 - x_1^2 + \frac{2}{\lvert J_2 \rvert} \sum_{j \in J_{12}} h(y_j)$ $where J_1 = \{ j \mid j\ is\ odd\ and\ 2 \le j \le n \}$ $and\ J_2 = \{ j \mid j\ is\ even\ and\ 2 \le j \le n \}$ $y_i = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n} \right), \ j = 2, \ldots, n.$ *and* $h(t) = \frac{\lvert t \rvert}{1 + e^{2 \lvert t \rvert}}$ *Its PF is* $f_2 = 1 - f_2^2, \quad 0 \le f_1 \le 1.$ *Its PS is* $x_j = \sin\left(6\pi x_1 + \frac{j\pi}{n} \right), \ j = 2, \ldots, n. \ 0 \le x_1 \le 1$

UF5	2	$f_1 = x_1 + \left(\frac{1}{2N} + \varepsilon\right)\left\|\sin(2N\pi x_1)\right\| + \frac{2}{\|J_1\|}\sum_{j\in J_1} h(y_j)$ $f_2 = 1 - x_1 + \left(\frac{1}{2N} + \varepsilon\right)\left\|\sin(2N\pi x_1)\right\| + \frac{2}{\|J_2\|}\sum_{j\in J_2} h(y_j)$ $where J_1 = \{j \| j \text{ is odd and } 2 \le j \le n\}$ $and\ J_2 = \{j \| j \text{ is even and } 2 \le j \le n\}. N \text{ is int}. \varepsilon \rangle 0$ $y_i = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2,\dots,n.$ $and \quad h(t) = 2t^2 - \cos(4\pi t) + 1$ *Its PF has* $2N+1$ *Pareto optimal solutions* $\left(\frac{i}{2N}, 1 - \frac{i}{2N}\right),\quad i = 0,1,\dots,2N.\ \ N = 10,\ \ \varepsilon = 0.1$
UF6	2	$f_1 = x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \varepsilon\right)\sin(2N\pi x_1)\right\} + \frac{2}{\|J_1\|}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20 y_j \pi}{\sqrt{j}}\right) + 2\right)$ $f_2 = 1 - x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \varepsilon\right)\sin(2N\pi x_1)\right\} + \frac{2}{\|J_2\|}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20 y_j \pi}{\sqrt{j}}\right) + 2\right)$ $where\ J_1 = \{j \| j \text{ is odd and } 2 \le j \le n\}$ $and\ J_2 = \{j \| j \text{ is even and } 2 \le j \le n\}$ $and \quad y_i = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2,\dots,n.$ *Its PF consist of one isolated point* $(0,1)$ *and* $N = 2$ *disconected parts :* $f_2 = 1 - f_1,\ \ f_1 \in \bigcup_{i=1}^{N}\left[\frac{2i-1}{2N}, \frac{2i}{2N}\right]$
UF7	2	$f_1 = \sqrt[5]{x_1} + \frac{2}{\|J_1\|}\sum_{j\in J_1} y_j^2$ $f_2 = 1 - \sqrt[5]{x_1} + \frac{2}{\|J_2\|}\sum_{j\in J_2} y_j^2$ $where\ J_1 = \{j \| j \text{ is odd and } 2 \le j \le n\}$ $and\ J_2 = \{j \| j \text{ is even and } 2 \le j \le n\}$ $and \quad y_i = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2,\dots,n.$ *Its PF is* $\quad f_2 = 1 - f_1,\ \ 0 \le f_1 \le 1$
UF8	3	$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\|J_1\|}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$ $f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{\|J_2\|}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$ $f_3 = \sin(0.5x_1\pi) + \frac{2}{\|J_3\|}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$ where $J_1 = \{j \| 3 \le j \le n$, and $j - 1$ is a multiplication of $3\}$, $J_2 = \{j \| 3 \le j \le n$, and $j - 2$ is a multiplication of $3\}$, $J_3 = \{j \| 3 \le j \le n$, and j is a multiplication of $3\}$.

		Its PF is $f_1^2 + f_2^2 + f_3^3 = 1, 0 \le f_1, f_2, f_3 \le 1$.
UF9	3	$f_1 = 0.5[max\{0,(1+\varepsilon)(1-4(2x_1-1)^2)\}+2x_1]x_2 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$ $f_2 = 0.5[max\{0,(1+\varepsilon)(1-4(2x_1-1)^2)\}-2x_1+2]x_2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$ $f_3 = 1-x_2 + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$ where $J_1 = \{j \mid 3 \le j \le n$, and $j-1$ is a multiplication of $3\}$, $J_2 = \{j \mid 3 \le j \le n$, and $j-2$ is a multiplication of $3\}$, $J_3 = \{j \mid 3 \le j \le n$, and j is a multiplication of $3\}$, and $\varepsilon = 0.1$ ε can take any other positive values. The PF has two parts. The first part is $0 \le f_3 \le 1,$ $0 \le f_1 \le \frac{1}{4}(1-f_3),$ $f_2 = 1 - f_1 - f_3;$