A COMPARATIVE STUDY OF DIVERSITY PRESERVATION TECHNIQUES IN MULTIOBJECTIVE EVOLUTIONARY ALGORITHMS

Arun Anand Sadanandan

Thesis submitted to the University of Nottingham
for the degree of Master of Philosophy

August 2007

Abstract

Most real world applications found in today's world necessitate dealing with certain common issues. These competing, often conflicting problems have kept researchers around the globe inquisitive and interested over the years and continue to do so, attributing to several open questions in the area. These problems, which deal with two or more objectives and invariably, involve large and complex search spaces, are referred to as multi-objective optimization problems (MOP's). Although several traditional methods has been put forth and tested, evolutionary algorithms is being reckoned to be one of the approaches that provide efficient and effective solutions to these challenging problems, mainly because of its ability to deal with problems that are multi-objective in nature. These algorithms are, naturally termed as multi-objective evolutionary algorithms.

Evolutionary algorithms are classified into three major forms: Genetic Algorithms (GA), Evolutionary programming (EP) and Evolutionary strategies (ES). Owing to the popularity of this area of research, several new approaches based on evolutionary techniques has evolved over the years. Additionally, many successful attempts towards improvement of these existing methods have emerged too. Moreover, other nature inspired approaches like particle swarm optimization and immune systems are widely researched as well. This thesis attempts to summarize and classify information on these biological inspired approaches, highlighting the importance of analyzing the research techniques followed by them thereby motivating researchers to come up with novel ideas for exploiting the search capabilities of these algorithms. A comparative analysis and study of the main algorithms are also provided based on diversity measures, along with their advantages and disadvantages and application areas. New approaches are proposed through hybridization methods on diversity techniques in multi-objective algorithms. A software toolbox for MOEAs is also developed. Finally, future development in this area and potential path for further research is addressed.

Acknowledgements

I would like to take this opportunity to thank everyone who painstakingly guided me during this work. I wish to express my sincere thanks and gratitude to my supervisors Dr. Nasreddine Hallam and Dr. Payam Barnaghi, for their excellent guidance, patience and support throughout my period of study. I am also grateful to Dr. Dickson Lukose for his constant motivation.

I would also like to thank several of my friends for their whole hearted support and helping with the preparation of this thesis, especially Thamil, Vijay and Khadir.

I wish to thank my parents and brother for their love and encouragement the entire time. And last but not least I would like to thank the God almighty for everything.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

This work aspires to make a thorough comparative analysis and study of nature inspired algorithms such as evolutionary algorithms, particle swarm optimization, artificial immune systems etc. in the context of multi-objective optimization problems. Furthermore two new methodologies are suggested by making use of hybridization techniques. This section introduces an overview of multi-objective optimization problems, brief introduction to the nature inspired algorithms studied in this thesis, problem definition, research goals and methodology, and structure of the thesis in general.

1.1 Multi-objective Optimization Problems (MOP)

Most problems in nature require managing several objectives which are mostly independent of each other. Consider a simple case of designing an automobile; the general objectives should be to minimize factors like cost and weight while maximizing performance, fuel efficiency or the like. Problems with more than one objective (often conflicting, as in the previous example) are commonly referred to as multi-objective optimization problems and are encountered in many real world applications, be it nature, science or business.

1.1.2 MOP Definition

Multi-objective optimization (also called multi-criteria optimization, multi-performance or vector optimization) can be defined as the problem of finding [16]:

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

Formally it is stated as follows [16]:

Find the vector $x^* = [x^*_1, x^*_2, \ldots\ldots, x^*_n]^T$ which will satisfy the *m* inequality constraints:

$$g_i(x) \geq 0 \quad i = 1,2,\ldots,m \tag{1}$$

the p equality constraints

$$h_i(x) \geq 0 \quad i = 1,2,\ldots,p \tag{2}$$

and optimizes the vector function

$$f(x) = [f_1(x), f_2(x), \ldots, f_k(x)]^T \tag{3}$$

where $x = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables.

In other words the aim is to determine from among the set F of all the numbers which satisfy (1) and (2) the particular set $x^*_1, x^*_2, \ldots\ldots, x^*_n$ which yields the optimum values of all the objective functions.

Thus the general form of a constrained multi-objective optimization problem is as follows:

Without loss of generality, in the case of a minimization problem to minimize means [17]

1) all objective functions are simultaneously minimized
2) the objectives are at least partly conflicting with each other, and
3) there does not exist any single solution that is optimal with respect to every objective function.

As is often the case with multi-objective optimization problems, there is no single optimal point or solution. Conflicting objectives demand a set of solutions which are essentially "tradeoff" solutions. These are referred to as *Pareto Optimal Solutions* and are the desired solution set of the MOPs.

We say that a vector of decision variables $x^* \in F$ is *Pareto optimal* if there does not exist another $x \in F$ such that $f_i(x) \leq f_i(x^*)$ for all $i = 1,\ldots,k$ and $f_j(x) < f_j(x^*)$ for at least one j [16].

In words this definition [16] says that x^* is Pareto Optimal if there exists no feasible vector of decision variables $x \in F$ which would decrease some criterion without a simultaneous increase in at least another criterion. The vectors x^* corresponding to the solutions included in the Pareto optimal set are called *non-dominated.* The plot of

the objective functions whose non-dominated vectors are in the Pareto optimal set is called the *Pareto front.*

A point $x^* \in F$ is a *weakly non-dominated solution* if there is no $x \in F$ such that $f_i(x) < f_i(x^*)$, for $i = 1, \ldots, n$.

A point $x^* \in F$ is a *strongly non-dominated solution* if there is no $x \in F$ such that $f_i(x) \leq f_i(x^*)$, for $i = 1, \ldots, n$ and for at least one value of i, $f_i(x) < f_i(x^*)$.

Thus, if x^* is *strongly non-dominated*, it is also *weakly non-dominated*; but the converse is not necessarily true[9]. As explained in [9], Non-dominated solutions for the bi-objective case can readily be represented graphically by passing into the objective function space {f1(x), f2(x)}. The so-called *minimal curve* corresponds to the locus of strongly non-dominated points, and the *weakly minimal curve* to the locus of weakly non-dominated points [9]. These two curves are sketched in Figure 1 for a simple bi-objective problem [9].

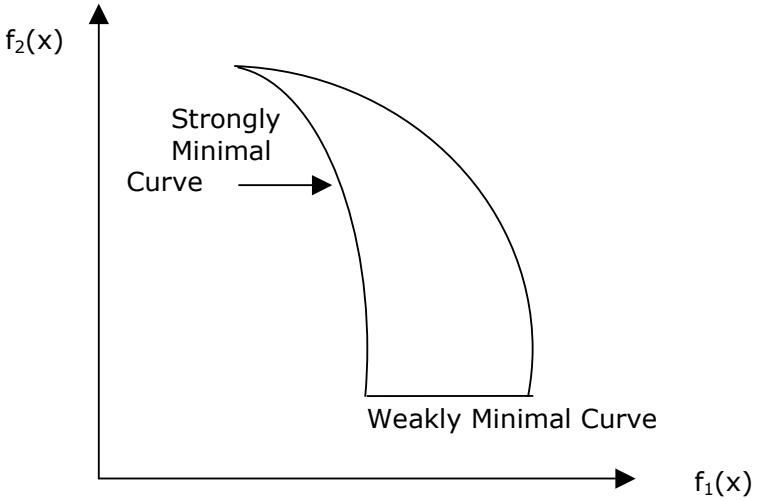


Figure 1: Weakly and strongly non-dominated curves on the bi-objective case (Source: Coello [9]).

1.2 Evolutionary Algorithms

The ever increasing need of effective solutions for multi-objective problems has focused the attention on various evolutionary methods over time. The main reason for preferring these approaches is the fact that multi-objective problems have to deal with a population of solutions as opposed to the single objective problem wherein the idea is to obtain a single solution as result. Evolutionary algorithms are proven to be an ideal technique in working with a population of solutions with the ability to come up with multiple solutions in a single simulation run. Consequently a number of approaches based on EAs have emerged [3], [4], [5], [6], [7], [8]. Evolutionary algorithms are heuristic search techniques that adopt a selection mechanism inspired on the *survival of the fittest* principle from Charles Darwin's evolutionary theory.

Ever since Schaffer (1984) [49] came up with research work in multi-objective genetic algorithms, active and persistent research has been performed in the field followed by Fourman (1985) [50], and Goldberg (1989) [20] whose study of non dominated genetic algorithm has played a pivotal role in further advancement in this field. A number of different EA implementations were proposed in the years 1991-2002 (Kursawe 1991 [41], Hajela and Lin 1992 [48]; Horn, Nafpliotis, and Goldberg 1994 [8]; Srinivas and Deb 1995 [4], E. Zitzler and L. Thiele 1999, K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan 2002 [5], E. Zitzler, M. Laumanns, and L. Thiele 2002 [6]).

The primary goal of every multi-objective evolutionary algorithm is to obtain a solution set that is as close as possible to the Pareto front much often called the true Pareto front. Although the true Pareto front is a hypothetic set of values most of the time and will be unknown in many real world scenarios, researchers make use of various standardized test methodology and performance metrics to evaluate the success of a Pareto front obtained in algorithm runs[10], [11]. In recent years, researchers have investigated particular topics of evolutionary multi-objective search, such as convergence to the Pareto-optimal front, divergence, niching, and elitism which are integral elements to be considered in modern evolutionary multi-objective approaches.

1.3 Particle Swarm Optimization (PSO)

1.3.1 Overview

A relatively new optimization technique by J. Kennedy and R. Eberhart (1995) is yet another method based on a biological approach [12]. Introduced as a method to facilitate single objective optimization, PSO is fast gaining popularity in solving multi-objective problems as well. Particle Swarm Optimization is a swarm intelligence method that models social behavior to guide swarms of particles towards the most promising regions of the search space [12]. Like evolutionary approaches PSO also maintains a population of solutions and individuals are represented using binary or floating point encoding similar to Evolutionary Strategies.

In PSO, the population dynamics simulate the behavior of "bird's flock", where social sharing of information takes place and individuals profit from the discoveries of previous experience of all other companions during the search of food [13]. The companions denoted as *particles* perform search in the population or *swarm*, for efficient results. The members of the population adjust their positions or parameters during the optimization run based on their previous experiences as well as the flying experiences of other members of the flock. i.e. the best evaluated individual found so far by the optimization process called *gbest* and the best evaluated individual found previously by the same individual denoted as *pbest.* A local individual may be selected for each swarm member, however these lbest individuals may all also be non-dominated (representing local areas of the estimated Pareto front maintained by the swarm), making them all also gbest [14].

1.3.2 PSO Definition

To define the operation of a PSO a fixed population of solutions is used, where each solution (or particle) is represented by a point in N-dimensional space. The i^{th} particle is commonly represented [13], [14], [15] as $X_i = (x_{i1}, x_{i2},\ldots.., xi_{iN})$, and its performance evaluated on the given problem and stored. Each particle maintains knowledge of its best previous evaluated position, represented as $P_i = (p_{i1}, p_{i2}\ldots\ldots,p_{iN})$, and also has knowledge of the single global best solution found so far, in the traditional uni-objective application indexed by g. The rate of position change of a particle then depends upon its previous local best position and the global best, and

its previous velocity. For particle i this velocity is $V_i = (v_{i1}, v_{i2}, \ldots\ldots, v_{iN})$. Hence the velocities of particles are determined as:

$$V_{i,j} = w\, v_{i,j} + c_1 r_1 (p_{i,j} - x_{i,j}) + c_2 r_2 (p_{g,j} - x_{i,j}) \quad (4)$$

$$x_{i,j} = x_{i,j} + X v_{i,j}, \qquad j = 1,\ldots,N \quad (5)$$

where w, $c_{1,}$ $c_{2,\, X \geq 0}$.w is the inertia weight of a particle, c1 and c2 are constraints on the velocity toward global and local best, X is a constraint on the overall shift in position, $r_{1,}$ r_2 are random numbers within the range [0,1].

[13] states that Equation (4) determines the i^{th} particle's new velocity as a function of three terms: the particle's previous velocity; the distance between the best previous position of the particle and its current position, and finally; the distance between the swarm's best experience (the position of the best particle in the swarm) and the i^{th} particle's current position. Then, according to Equation (2), the i^{th} particle "flies" towards a new position. In general, the performance of each particle is measured according to a fitness function, which is problem-dependent [13].

1.4 Research Problem and Objectives

With multi-objective evolutionary algorithms being a rapidly evolving field of research with new methodologies being introduced, several optimization techniques found for improvement of these approaches, considerable number of research publications released in regular intervals, the need for an updated analytical work is essential. Recently, lot of focus in this field of research has been on improvement of techniques used in MOEA's mainly convergence and diversity. Achieving these would be the fundamental goals of any MOEA design; minimizing the distance of the generated solutions to the Pareto front (convergence) and maximizing the spread of the achieved Pareto set approximation (diversity). Diversity plays a crucial part in producing successful results because it ensures effective search in the most of the areas of the objective space, thereby avoiding the problem of potentially good

solutions being ignored, as well as providing the decision maker with efficient set of solutions.

Traditionally, diversity was achieved through fitness sharing mechanisms suggested by Goldberg [19], [20]. But modern MOEAs are benefited by the use of several archive-updation procedures in achieving this goal. In this thesis we perform a study into these diversity preservation methods used in Non Dominated Sorting Genetic Algorithm (NSGA II), Pareto Archived Evolutionary Strategies (PAES), Strength Pareto Evolutionary Algorithm (SPEA 2) and Pareto Potential Regions Evolutionary Algorithm (PPREA). By doing this, improved versions of NSGA II and SPEA2 are implemented using hybridization methods. This is achieved by using alternate diversity measures on these algorithms. The improved versions are tested using standardized test functions and performance metrics and evaluated. This being the primary motivation of the thesis, it is also intended to provide detailed comparative analysis of several MOEAs. A multi-objective optimization toolbox is also designed in this work.

Thus the research goals are broadly classified into three stages:

1. A comprehensive comparative analysis of biologically inspired algorithms, in terms of features such as convergence and diversity, and arrive at a conclusion on the efficiency and effectiveness of these. Diversity techniques are given more importance in doing so.
2. Based on diversity handling techniques, introduce hybridization methods for MOEAs and perform thorough and accurate analysis and performance comparison based on test problems and performance metrics.
3. Design and develop a toolbox with extended functionalities aiding research in the field of multi objective evolutionary algorithms.

1.5 Thesis Structure

Chapter 2 presents the literature review of this work with classification and overviews of evolutionary algorithms and introduction to definitions and terms. Chapter 3 discusses the various test functions and performance metrics and their classification. Definitions and depiction of true Pareto front representation of these problems are defined too. In Chapter 4, a new tool box for MOEA's is introduced and discussed. Chapter 5 details the research methodology and the implementation of the new methods. Chapter 6 provides a summary of the results and discusses the findings of this research. Chapter 7 provides the conclusion and discusses the future scope of research.

Chapter 2

Multi-Objective Evolutionary Algorithms

In this chapter the basic principles of evolutionary algorithms and its importance in multi-objective optimization problems are reviewed. The general structure and mechanisms followed by the EA is detailed in section 2.1, followed by overviews of key concepts in EAs addressed in section 2.2. In section 2.3, some of the popular Multi-objective Evolutionary Algorithms (MOEAs) are reviewed.

2.1 Classification of Evolutionary Algorithms

Evolutionary algorithms, a subset of evolutionary computation, are computer-based problem solving systems that models evolutionary processes. These computational models act as primary elements of the design and implementation of an evolutionary algorithm. Although the origins of evolutionary inspired algorithms for optimization and machine learning can be traced to as early as 1950s [20], it has gained considerable popularity over the last few decades.

Evolutionary algorithms can be broadly classified into three mainstream instances:

- Genetic Algorithms
- Evolutionary Programming
- Evolutionary strategies

Other evolutionary approaches like genetic programming, classifier systems and several hybridization methods have also evolved. Despite of different evolutionary algorithms being proposed, all of them are similar in their basic properties.

- Evolutionary algorithms operate on a population of individuals thereby incorporating a collective learning process. Each individual represents and encodes a search point in the space of potential solutions to a given problem.

- Quality of individuals is evaluated by assigning a quality measure to them, referred to as the fitness of the individuals. Then a selection process is performed in such a way that the fitter individuals have a higher probability of taking part in the search in future generations.

- New candidates for the coming generations are selected by sets of processes which model natural process such as recombination and mutation.

Thus in evolutionary algorithms, the solutions are termed as individuals and the set of individuals as population, borrowing from natural terms. The following algorithm represents a general evolutionary algorithm. A population *P* of individuals is initialized and then evolved from generation *t* to generation *t*+1 by repeated application of fitness evaluation, selection, recombination, and mutation. In the selection process, which can be either stochastic or completely deterministic, individuals with lower fitness are removed from the population, while fitter individuals have higher chance of reproduction. Recombination and mutation aim at generating new solutions within the search space by the variation of existing ones. The crossover operator takes a certain number of parents and creates a certain number of children by recombining the parents. To mimic the stochastic nature of evolution, a crossover probability is associated with this operator. Similarly, the mutation operator alters individuals according to a mutation probability. The crossover and mutation operations are performed on individuals, i.e., in individual space, and not on the decoded decision vectors. Based on these concepts, natural evolution is simulated by an iterative computation process. The various steps involved in the algorithm are explained below.

2.1.1 A General Evolutionary Algorithm

Algorithm 1: General Evolutionary Algorithm

Input: *N (population size)*
T (maximum number of generations)
pc (crossover probability)
pm (mutation rate)

Output: *A (nondominated set)*

Step 1: ***Initialization**: Set* $P_0 = \ulcorner$ *and* $t = 0$. *For* $i = 1, \ldots, N$ *do*
a) Choose $i \in I$ *according to some probability distribution.*
b) Set $P_0 = P_0 + \{i\}$.

Step 2: ***Fitness assignment**: For each individual* $i \in P_t$ *determine the encoded*

decision vector $x = m(i)$ *as well as the objective vector* $y = f(x)$ *and calculate the scalar fitness value* $F(i)$.

Step 3: ***Selection**: Set* $P' = \Gamma$. *For* $i = 1, \ldots, N$ *do*
a) Select one individual $i \in P_t$ *according to a given scheme and based on its fitness value* $F(i)$.
b) Set $P' = P' + \{i\}$.
The temporary population P' is called the mating pool.

Step 4: ***Recombination**: Set* $P'' = \Gamma$. *For* $i = 1, \ldots, N/2$ *do*
a) Choose two individuals $i, j \in P'$ *and remove them from* P'.
b) Recombine i and j . The resulting children are $k, l \in I$.
c) Add k, l to P'' with probability p_c. *Otherwise add* i, j *to* P''.

Step 5: ***Mutation**: Set* $P''' = \Gamma$. *For each individual* $i \in P''$ *do*
a) Mutate i with mutation rate p_m. *The resulting individual is* $j \in I$.
b) Set $P''' = P''' + \{j\}$.

Step 6: ***Termination**: Set* $P_{t+1} = P'''$ *and* $t = t + 1$. *If* $t \geq T$ *or another stopping criterion is satisfied then set* $A = p(m(P_t))$ *else go to Step 2.*

2.1.1.1 Input and Output Parameters

Input parameters are:

- **N** is the total number of individuals in the population.
- **T** is the maximum number of generations.
- **Pc** is the probability of crossover.
- **Pm** is the probability of mutation

Output parameter is the Pareto optimal set stored in **A**.

2.1.1.2 Initialization

The first step of an EA is to initialize the population in random order. The set Pt gives the population at generation t and is generally stored as a linear list. The individuals in the population are encoded using binary representation, real representation or other schemes such as graphs and trees. In binary encoding, individuals are referred to as *chromosomes* and each bit a *gene.*

2.1.1.3 Fitness Assignment

The fitness of an individual determines its 'quality' in terms of the problem in hand. In most cases it is a scalar vector produced by the evaluation of an objective function although this may vary with different MOEAs. Currently there are several

fitness assignment strategies. MOEAs differ in accordance to the fitness assignment and selection methods. They could be classified as:

Classical aggregation techniques: The objectives are aggregated into a single parameterized objective. Each objective function is multiplied with its corresponding weight, and they are summed up to obtain the fitness value that is to be assigned to the individual. The weighted sum aggregation approach is an example of this type. The parameters of this function are not changed for different optimization runs, but instead systematically varied during the same run. Some approaches such as (Hajela and Lin 1992 [48]), for instance, use the weighting method. Since each individual is assessed using a particular weight combination (either encoded in the individual or chosen at random), all members of the population are evaluated by a different objective function. Hence, optimization is done in multiple directions simultaneously. However, these approaches experienced difficulties when a non-convex pareto front is encountered.

Criterion-based methods: Instead of combining the objectives into a single scalar fitness value, this class of MOEAs switches between the objectives during the selection phase [22]. Each time an individual is chosen for reproduction, potentially a different objective will decide which member of the population will be copied into the mating pool.

Pareto-based Selection: This popular method, suggested by Goldberg [20] calculates the individuals fitness based on Pareto dominance. First all nondominated individuals are assigned rank one and temporarily removed from the population. Then, the next nondominated individuals are assigned rank two and so forth. Finally, the fitness value is determined by the rank of an individual. This way the fitness is related to the whole population, unlike other approaches where an individual's raw fitness value is calculated independent of other individuals. Several of the successful algorithms developed has benefited by Pareto based fitness assignment scheme [4], [8], [27].

2.1.1.4 Selection

Selection plays a crucial part in the outcome of an MOEA. Selection operator selects the individuals for reproduction on the basis of their fitness values, thereby selecting individuals with higher quality for survival. There are different kinds of selection

operators, of which *Roulette wheel selection* and *Tournament selection* are popular. In Roulette wheel selection, each individual is assigned a proportion of the roulette wheel equal to the ratio of its fitness to the sum of the entire population's fitness, whereas in Tournament selection, n individuals are chosen at random and the best one is selected. Constraint handling is also done in the selection step.

2.1.1.5 Recombination

Recombination provides variation to the individuals in the population thereby allowing the algorithm to explore new regions of the search space. This is done through crossover and mutation operators.

- Crossover: There are several crossover methods. Their usage is often dependent on the type of encoding used in the individuals. For example for binary and integer encoding, single point, *n* point and uniform crossover are used whereas for real representations, simulated binary crossover operators are used. Crossover is performed only with a certain probability.

    - Single point crossover: A crossover point on the bit string is selected by random and all the elements following this point are interchanged.
    - *n* point crossover: Here *2n* crossover points are selected randomly and the elements between these points are interchanged.
    - Uniform crossover: This operator randomly chooses different bits from each parent, with equal probabilities.

- Mutation: Mutation operator is used to introduce completely new individuals suddenly into the population to improve the search. This is usually done with a lesser probability. In case of binary encoded individuals, mutation changes a bit from 0 to 1 and vice versa.

2.1.1.6 Termination

An EA will stop when the termination criteria is met. This happens usually when the maximum number of generations has been reached or when there is no change in the population for several generations.

2.1.1 Genetic Algorithms

Genetic Algorithms (GAs) were invented and developed by Holland [21]. Holland's original goal was to formally study the phenomenon of natural adaptation and to develop ways in which its mechanism might be imported into computer systems. GAs were presented as an abstraction of biological evolution and derived its behavior from a genetic/evolutionary metaphor.

Traditionally, GAs use binary representation for the individuals (chromosomes or structures). Recently, however, many applications have focused on other representations such integers, real-valued vectors, graphs (neural networks), Lisp expressions, and ordered lists.

Selection is a probabilistic function based on relative fitness. With this selection method, known as fitness-proportional selection, the expected number of times an individual will be selected to reproduce is the individual's fitness divided by the average fitness of the population. A simple method of implementing fitness-proportional selection is roulette-wheel sampling [16]. The number of offspring created is the same as the number of parents μ. Later, in the survivors selection step, the μ newly created offspring will replace the μ parents in the population. This form of selection is not elitist.

Offspring are created by recombination (crossover) of parent individuals with probability *pc*. After that, mutation is applied with a very small probability *pm* per bit. In its initial conception, GAs emphasize recombination (crossover) as the primary search operator and apply mutation solely as a "background operator". Interest in mutation has increased recently, partly due to the influence of Evolution Strategies and Evolutionary Programming.

2.1.2 Evolutionary Strategies

Evolution strategies (ESs) were developed by Rechenberg [12], using selection, mutation, and a population of one parent and one offspring. Schwefel [13] introduced recombination and populations with more than one individual, and compared ESs with more traditional optimization techniques.

Evolution strategies typically use real-valued, vector representations. Individuals to be parents are selected randomly from a uniform distribution. The number of offspring λ created is greater than the number of parents μ. The selection of survivors is deterministic and is implemented in one of two methods. The first method selects the best μ out of λ offspring and replaces the parents with these newly created individuals. In other words, only the best μ offspring are allowed to survive. This method is known as a (μ,λ) selection strategy. The second method selects the best μ individuals among μ parents and λ offspring. Thus, in this method both the best parents and offspring are allowed to survive. This second method is known as a $(\mu+\lambda)$ selection strategy. Both methods belong to the kind of extinctive (truncation) selection methods. $(\mu + \lambda)$ selection is elitist but (μ,λ) selection is not.

Offspring are created by recombination (when $\mu > 1$) of parent individuals followed by mutation. A variety of different recombination mechanisms are currently used in ESs and the operators are sexual and panmictic. In sexual operators, recombination acts on two randomly chosen parent individuals. Conversely, in panmictic operators, one parent is randomly chosen and held fixed while for each component of its vectors the second parent is randomly chosen anew from the population. Mutation perturbs the individuals using a normal distribution with expectation zero. In ESs considerably effort has been put on (self) adapting the mutations during the run of the algorithm. ESs allow each individual (or each variable within the individual) to have adaptive mutation rates that are normally distributed with a zero expectation.

ESs emphasizes recombination and mutation as essential operators for searching simultaneously in the variables space and in the strategy parameters space (self-adaptation).

2.1.3 Evolutionary Programming

Evolutionary programming (EP) was developed by Fogel et al. [14]. EP arose from the desire to generate machine intelligence using selection and mutation to evolve finite-state machines.

EP traditionally has used representations for the individual that are tailored to the problem domain. In the case of finite-state machine applications, the individuals within the population are represented as graphs. For other applications, appropriate representations such real-valued vectors and ordered lists are used.

Selection is a probabilistic function based on tournament. The number of offspring created is the same as the number of parents μ. In the survivors selection step, μ individuals are chosen from the 2μ (parents and offspring) individuals. This form of selection is elitist and can be considered to be a $(\mu + \mu)$ selection strategy.

Offspring are created by mutation of parent individuals. The form of mutation is based on the representation used, and similar to ESs is often (self) adaptive. For real valued optimization problems, for example, it works with normally distributed mutations with expectation zero and extends the evolutionary process to the strategy parameters (self-adaptation). The forms of mutation used are usually quite flexible and can produce perturbations similar to recombination, if desired. However, EP emphasizes mutation and does not incorporate the recombination of individuals. The justification for this is that in EP each individual is usually viewed as the analog of a species, and there is no sexual recombination between species.

2.2 MOEA design challenges

Success of an MOEA relies mainly on two factors:

- Convergence: The approximation set should contain solutions whose corresponding objective vectors are close to the true Pareto front. This is achieved by assigning scalar fitness values to solutions in the presence of multiple optimization criteria.
- Diversity: The obtained non dominated set should contain solutions, which are evenly distributed i.e. they should maintain uniform distance, usually in the objective space, but it is preferred that the diversity is extended to the objective space as well.

Computational efficiency while achieving these goals is a factor to be considered as well. Finally, another issue that addresses both of the goals of MOEA is elitism.

2.2.1 Convergence

The selection methods that operate giving preference to solutions that are locally non-dominated solutions over their dominated counterparts help the population to evolve towards globally optimal front. These methodologies are based on a partial ordering, or ranking, of the population. As stated by Zitzler (2002) [26], these schemes make use of the following information for each solution that can be drawn from the current population:

- Dominance rank. The number of solutions in the population that dominate the solution under consideration.
- Dominance count. The number of solutions in the population that are dominated by the solution under consideration.
- Dominance depth. The rank of the solution in the non-dominated sorting of the population.

Non-dominated sorting was the original Pareto-based EA approach proposed by Goldberg [20]. Here, he locally non-dominated solutions in the population are identified, assigned rank 0, and are removed temporarily from the population. In the remaining population, the new locally non-dominated solutions are identified, assigned rank 1, and are removed. This process is continued until all solutions have been assigned ranks.

Fonseca and Fleming used a dominance ranking method called Pareto-based ranking, which was implemented in the MOGA[27]. Later Zitzler and Thiele's [28] strength-based approach made use of both dominance rank and dominance count. A modified version has also been proposed in the SPEA2 [6]. A sample-based approximation to a Pareto-based ranking of the entire population was proposed by Horn and Nafpliotis [29].

In methods used by Fonseca and Fleming [27] and Srinivas and Deb [4] the rank values are assigned to fitness values, typically through linear transformation, and then proportional selection methods are applied thereby forming the mating pool. In SPEA and SPEA2, binary tournament selection operates directly on the strength-based fitness measure [2], [6] and on the dominance depth in Deb, Pratap, Agarwal and Meyarivan's NSGA-II [5]. However, in some methodologies, such as the PESA

family of algorithms [30], [31], selection operates purely on locally non-dominated solutions and thus no Pareto-based selection is required at this stage.
Elitism, which is discussed later in section 2.2.3, is also a required element for an EA to guarantee convergence in the limit to the global optimum,

2.2.2 Diversity

Diversity preservation has been given more and more importance as algorithm development in MOEAs progressed over the years. This could mainly be attributed to the ability of diversity methods in improving the quality of the non dominated set of solutions. Traditionally diversity has been achieved through fitness sharing and Pareto Niching functions, introduced by Goldberg [20]. However, with the introduction of new MOEAs to the community, the diversity methods also took new forms. This can be illustrated by referring to the diversity preservation techniques used in some of the widely used MOEAs in recent past such as SPEA2 [6], NSGAII [5], PAES [33] and others. NSGAII uses a Crowding distance estimate to perform diversity measures, while SPEA2 uses a truncation method based on the Nearest Neighbor density measure. PAES uses an adaptive Grid Algorithm to achieve diversity. These methods are discussed in detail in Chapter 5.

2.2.3 Elitism

During the optimization process, it could happen that potential good solutions are lost owing to random effects. This issue can be solved using an elitism approach. There are two possible ways of going about elitism. As mentioned in [22], combining parent and child population and applying a deterministic selection procedure, instead of changing the whole of the old population by the newly attained pool of individuals, is one way. The second and more popular method is use an external archive to store the best individuals found at each generation. The members of the archive usually take part in future selection procedures. An issue that has to be addressed in performing elitism is the amount of resources that could be used by the process. So usually a restriction measure is attached to the size of the archive. As and when the archive is overfilled, necessary measures are taken to decide whether to replace solutions from the archive, or to ignore the new solution. Most of the recent MOEAs make use of an external archive for storing elite individuals. The SPEA algorithm [3], for example, stores all non-dominated solutions separately from the active

population. The selection process involves both the archive and the population, with preference given to individuals in the elite set.

2.3 Review of Multi-objective Evolutionary algorithms

Contemporary MOEAs use selection and replacement based on multi-objective domination criterion. Examples of this approach are Fonseca and Fleming's MOGA [23], Horn et al.'s NPGA [25], Corne, et al.'s PESA [31], Zitzler et al.'s SPEA algorithms [3], [6] and Deb et al.'s NSGA-II [5]. All of these algorithms use niching to ensure that a diverse Pareto set is found, and all except one (the NPGA) use elite methods or an external storage to keep the best individuals found so far. Here we discuss some of the MOEAs relevant to this thesis.

2.3.1 Strength Pareto Evolutionary Algorithm (SPEA)

SPEA introduced by Zitzler and Thiele [3] is an MOEA which uses elitism with the concept of non-domination. As discussed in the previous section on elitism, this method uses an external population to store non-dominated individuals. The members of this external population also take part in recombination and selection operations. The number of non-dominated solutions determines the fitness values of individual in both the populations. The algorithm follows these steps. First a new population is produced by combining the external population and the regular one. Then, fitness values are assigned to each of the members based on how many solutions they dominate. Another factor that is considered here is to assign more fitness values to those solutions which are having more dominated solutions. This measure is called a strength value which is a measure of the number of solutions in the current population that is dominated by that solution in the external archive. To deal with crowding of individuals a clustering mechanism is used where solutions in a less crowded non-dominated front are retained. By using this method, diversity is achieved. But there are drawbacks in this method since a Pareto optimal solution in the external population may get replaced by a less efficient solution that is in a less crowded region.

2.3.2 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

The SPEA2 [6] is a modified version of the SPEA proposed by the same authors. The primary motivation for the design of this algorithm was to overcome weaknesses

found in its predecessor [6] in addition to the new strategies proposed. The new algorithm uses a fitness assignment method, whereby each member of the solution set is assigned fitness based on a value indicating the number of solutions which are dominated by that individual as well as the number of solutions which dominates the individual. It proposes a new method for diversity based on the k^{th} nearest neighbor method. Further the problems of boundary solutions being lost in SPEA is solved by using a new archive truncation method.

The fitness assignment in SPEA2 is the sum of the raw fitness measure and the distance found from the nearest neighbor estimation technique. The raw fitness of an individual is a measure of the sum of the strength values, i.e. the number of solutions that it dominates. The fitness assignment process is discussed in detail in the diversity preservation methods defined in section 5.1.2.

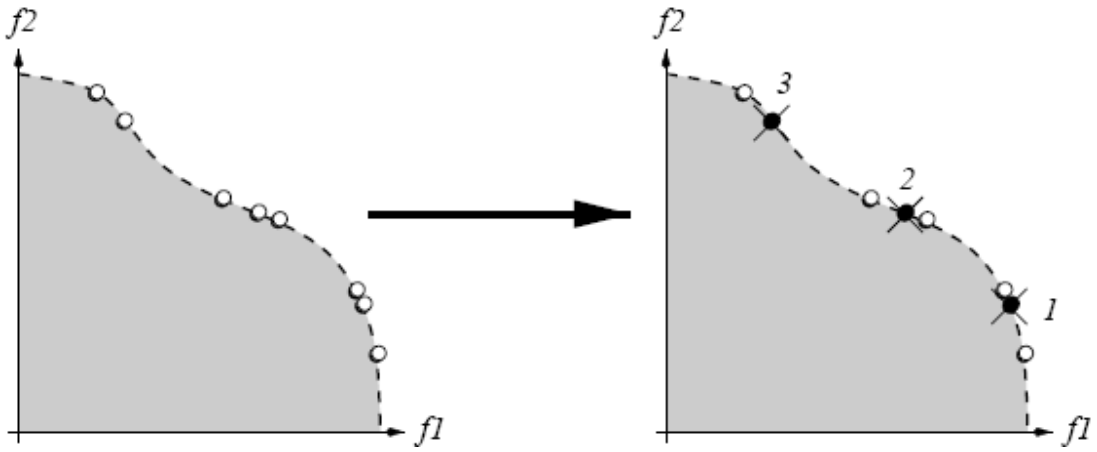


Figure 2: Archive truncation in SPEA2 (Source: Zitzler et. al [6])

The next stage is the environmental selection stage, where archive updation is achieved. This method improves upon the one used in SPEA such that the boundary solutions are preserved and would not take part in the truncation process. In SPEA2 the size of the archive is kept constant unlike SPEA. In this stage the archive is filled with non-dominated solutions. The updation process stops if the number of non-dominated individuals exactly fit the archive. In the event of the archive not being full, it is populated with dominated individuals from the previous archive. On the other hand if the number exceeds the size of the archive, truncation is performed until the required archive size is obtained. This is done with the help of the nearest neighbor method, the individual with the least distance value to another, is chosen at each stage for removal. This is illustrated in figure 3 taken from [6].

2.3.3 Non-Dominated Sorting Genetic Algorithm (NSGA-II)

NSGAII [5], which is an improved version of the NSGA [4], as the name suggests uses the strategy of sorting based on the level of non-domination. It uses elitism for preserving the best solutions, and has an explicit diversity preservation mechanism. The niching operation is performed without the explicit declaration of the sharing parameter. The book keeping strategy defined by [5] in this method ensures that the complexity of the algorithm is $O(MN^2)$, where M is the number of objectives and N is the size of the population. In this method, an external archive population P' is maintained, the contents of which are compared with all the members of the main population P one by one. On the event of a solution in P dominating one in P', the dominated member is removed from P'. If the solution in P is dominated by at least one member of P', then it is ignored, whereas if it is not dominated by any of the members then that solution gets added to the non-dominated set P'. This way the non-dominated list is produced for the entire population. After this the non-dominated sorting is performed by removing the non-dominated solutions from P and repeating the above procedure with the solutions stored in first front, second front and so on.

Following this step the well spread of solutions in the population is ensured by the diversity preservation method called crowding distance assignment. This is a density estimation metric which allows the comparison of solutions for the extent of proximity to other solutions. Then the solutions in the most crowded region is determined using a crowding comparison operator, which uses the crowding distance and domination ranking to find crowded regions. This diversity method is discussed in detail in section 5.1.3.

Thus in NSGA-II, the child populations Q_t is produced from the parent population P_t using selection, recombination and mutation operators. Then the two populations are combined together to produce R_t, which is of size twice the populations size N. After this the population R_t undergoes non-dominated sorting. This way a global non-domination check is achieved. Next P_{t+1} is filled based on the ranking of the non-dominated fronts. Since the combined population is twice the size of the population size N, all the fronts are not allowed to be used. Therefore a crowding distance sorting is performed in descending order and the population is filled. This for this new population P_{t+1} the whole process is repeated. The NSGAII process can be visualized in the figure 2 extracted from [5].

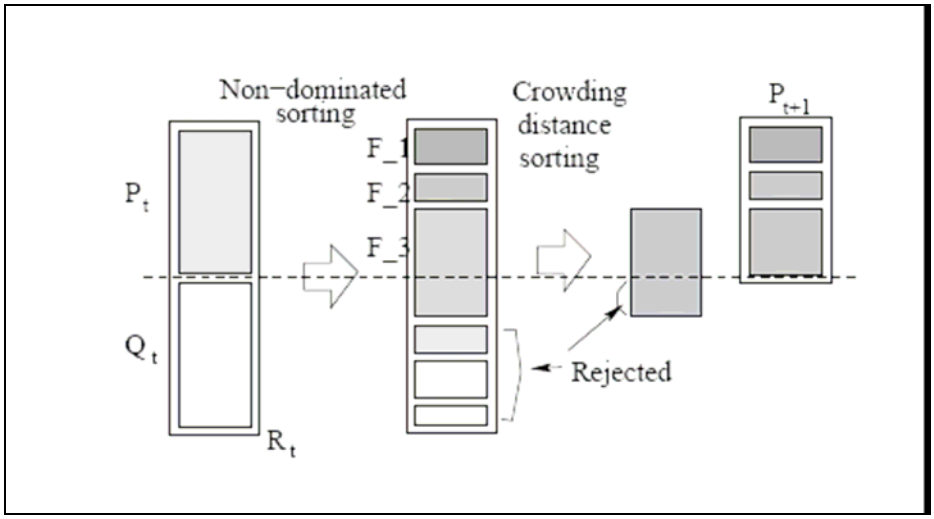


Figure 3: NSGAII process flow (Source: Deb et. al [5])

However NSGA II could have problems in convergence in certain scenarios where more than N population members of the combined population are found to be belonging to the non-dominated set, the selected is performed only based on crowding. This could potentially result in a Pareto optimal solution being omitted owing to a non optimal set which has better diversity metric value.

2.3.4 Pareto-Archived Evolution Strategy (PAES)

PAES introduced by Knowles and Corne [33] was first developed to find solutions using a local search method for multi-objective problems. It was mainly used for solving telecommunication routing optimization problems [46].

PAES acts as a multi-objective local search process which maintains a single parent solution and generates a single offspring on every iteration, through the process mutation. In this (1+1) evolution strategy, the offspring solution is compared to the parent and evaluated for domination. If the parent is dominated, then the offspring becomes the new parent and generation advances. Otherwise if the parent dominates the child, then the offspring solution is discarded and a new mutation solution takes part in the evolution. If neither dominates the other, then the selection is based on comparison with the individuals in the elite population. Two scenarios could arise from here. One, the offspring dominates one of the archive members. Here the offspring becomes the new parent and the archive is updated by removing the dominated solutions. On the other hand if the offspring does not

dominate any archive member, then a crowding check is performed after which the choice is made by identifying the proximity to other solutions in the archive. In the event of equal proximity values, random selection is performed to choose a parent.
Crowding is achieved using an adaptive grid algorithm method by recursively dividing the d dimensional objective space [46] where d is the depth parameter. The resultant n subspaces are updated dynamically. This method ensures low computational cost as compared to other methods. The essential preference of non-dominated solutions, the ability to converge quicker and the low computational head makes this method a popular algorithm for solving MOPs.

2.3.5 Pareto Potential Regions Evolutionary Algorithm(PPREA)

This MOEA proposed by Hallam [18] introduces a new fitness assignment scheme and archive updation procedure. In this method a chain of regions connecting successive points in the objective space is constructed. These regions, termed as *Potential Pareto Regions,* are dynamic regions within which any generated vector solution is automatically non-dominated with regard to all the current non-dominated solutions.

***Fitness assignment*:** The non-dominated set is sorted according to one objective and each pair of immediate neighbors delimit one PPR [18].The fitness values for the rest of the individuals in the population are assigned according to the distance of each individual from the PPR. The lower the fitness value of the individual, the better the individual is. All non dominated individuals are assigned with negative values. Additionally the fitness values of the non-dominated extreme solutions in the archive are calculated as the absolute value of twice the size of the largest PPR[18] and the rest of the archive members are assigned fitness values equal to the sum of the sized of two adjacent PPRs. The fitness values of all the other dominated members are determined by the Euclidean distances to the nearest PPRs. This way the non dominated members in the archive have better preference.

***Archive update procedure*:** Archive updation in PPREA is performed by rearranging the whole population including the archive and the rest of the population into a set of lists. The sorting is based on one of the objectives chosen. Then each of the non-dominated members are assigned with a list of points that they dominate. This list is sorted according to the fitness in ascending order. Then the archive is updated with the best individuals from the lists and subsequently by the next best individuals until the size criteria is met. In case the archive is full requires the entry of a new

individual, the removal process is performed on the most crowded region using the crowding dispersion technique.

Crowding dispersion method: This method allows computing the *degree* of crowding and the *extent* of distribution of each non-dominated solution in the archive. For each non-dominated vector a matrix is maintained which stores the two Euclidian distances to its immediate neighbors. The vectors in the archive are ordered according to one objective dimension and hence the immediate neighborhood is based on this order [18].

If the size of the archive is *n*, there is a need to compute *n* - 1 distances.

Based on this matrix, the two important indicators,

namely *crowding* and *dispersion* are computed as follows:

$crowding(i) = min(d_{hi}, d_{ij})$ (1)

$dispersion(i) = max(d_{hi}, d_{ij})$ (2)

Assume that *i* is the vector with the minimum crowding value and that $d_{hi} > d_{ij}$, then the vector *j*, the neighbor of *i*, has also the minimum crowding. Therefore the most crowded among these two vectors is the one with the minimum dispersion. The following table taken from [18] demonstrates the crowing dispersion indicators.

Efficient Solution	$Distance_{Neighbour1}$	$Distance_{Neighbour2}$
1 : first	-	$d_{first2nd}$
...	...	...
h	...	d_{hi}
i	d_{hi}	d_{ij}
j	d_{ij}	d_{jk}
...	...	...
n : last	d_{1last_last}	-

Table 1: PPRs dimensions used as crowing dispersion indicators (Source: Hallam [18]).

2.4 Summary

This chapter provides an introduction to the basic principles and classifications of Multi-objective Evolutionary algorithms. The major issues in the design of EAs such as diversity, convergence and elitism are addressed. Finally some of the popular MOEA methods are reviewed and highlighed. This chapter discusses the importance of fitness assignment and archive updation procedures used in Multi-Objective optimization problems.

Chapter 3

Test Problems and Performance Metrics

This chapter details the various test problems and performance metrics used for performing comparative analysis of evolutionary algorithms. The test problem definitions and graphs are represented in section 3.1 and the performance metrics are briefly reviewed in section 3.2.

3.1 Test Suite Problem

Test problems provide benchmarks for testing the various algorithms. Since most real world problems involve wide range of difficulty settings, the only way to decide upon the authenticity of a newly proposed algorithm or methodology is to agree upon a common set of benchmarking problems. Many test problems with varying difficulty settings and complex search spaces have been extensively used by the research community. Some of the desired factors in constructing the test problems are easiness of construction, scalability in terms of objective functions and decision variables, and the ability to simulate difficult scenarios, the occurrence of which is quite often in real life problems.

Usually the test problems are constructed in steps, where the Pareto optimal front is represented mathematically, then designing the objective search space with it and finally mapping the decision space into the objective space. In the next section some of the most popular test functions widely used in current research is discussed. Visual representation of the true Pareto front identified for these test problems are also featured.

3.1.1 ZDT Test Functions

Zitzler, Deb and Thiele [34] introduced six test functions which are widely used by researchers, owing to the ability to provide sufficient complexity in comparing different multi-objective optimizers. These test functions, which are commonly referred to as ZDT test functions are defined below.

As mentioned in [11] all of the six test functions defined follow the same structure with three basic functions.

$$\begin{aligned} &\text{Minimize } F(x) = (f_1(x_1), f_2(x_2)) \\ &\text{subject to } f_2(x) = g(x_2, ..., x_n)h(f_1(x_1), g(x_2, ..., x_n)) \\ &\text{where } x = (x_1, ..., x_n) \end{aligned}$$

(3.1)

The function f_1 is a function of the first decision variable only, *g* is a function of the remaining *n-1* variables and the parameters of *h* are the function values of f_1 and *g* [34]. The test functions vary in these three functions as well as in the number of variables m and the values associated to them.

3.1.1.1 ZDT1 Function

ZDT1 is a bi-objective MOP. It's a scalable problem in terms of the number of variables. By default ZDT1 uses 30 real variables. ZDT1 problem (shown in Equation 3.2) has a convex Pareto optimal front [34].

Definition:

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)\left[1-\sqrt{x_1 / g(x)}\right] \\ g(x) &= 1+9(\textstyle\sum_{i=2}^{n} x_i)/(n-1) \end{aligned} \qquad (3.2)$$

where n =30 and $x_i \in [0,1]$

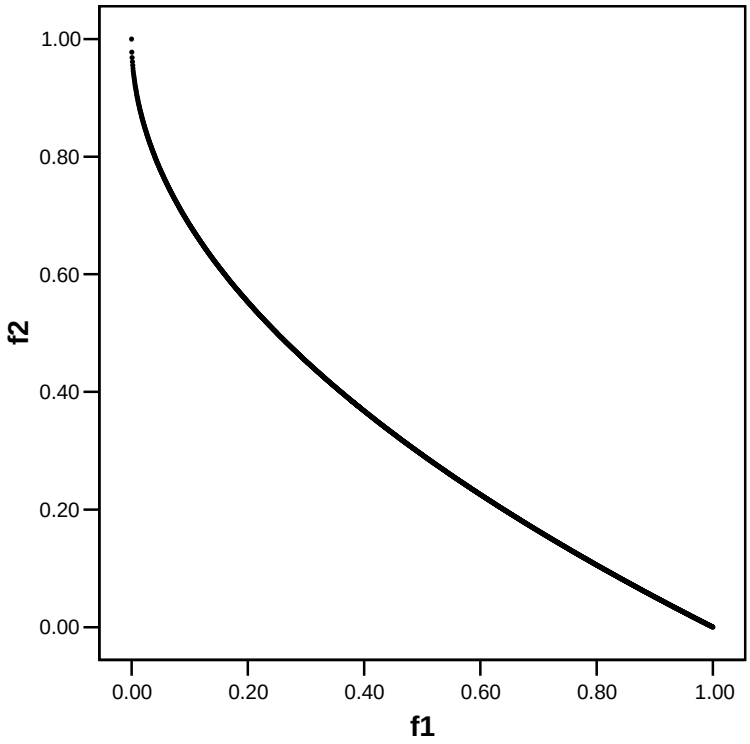


Figure 4: Pareto optimal front for ZDT1 function

3.1.1.2 ZDT2 Function

Like ZDT1, ZDT2 is a bi-objective MOP scalable in the number of variables. Also, by default, ZDT2 uses 30 real variables. ZDT2 problem (shown in Equation 3.3) has a non-convex Pareto optimal front [34].

Definition:

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)\left[1-\left(x_i / g\left(x\right)\right)^2\right] \\ g(x) &= 1+9\left(\sum_{i=2}^{n} x_i\right)/\left(n-1\right) \end{aligned} \tag{3.3}$$

where n =30 and $x_i \in [0,1]$

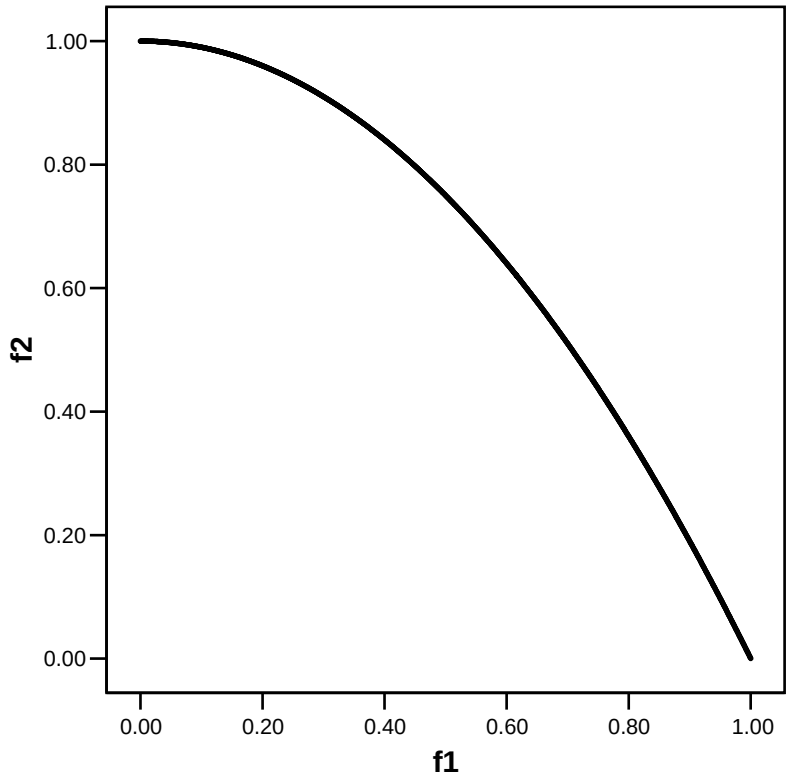


Figure 5: Pareto optimal front for ZDT2 function

3.1.1.3 ZDT3 Function

ZDT3 (shown in Equation 3.4) has a Non-convex and disconnected Pareto optimal front.

Definition:

$$
\begin{aligned}
& f_1(x) = x_1 \\
& f_2(x) = g(x)\left[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)}\sin\left(10\pi x_1\right)\right] \\
& g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)
\end{aligned}
\tag{3.4}
$$

where n =30 and $x_i \in [0,1]$

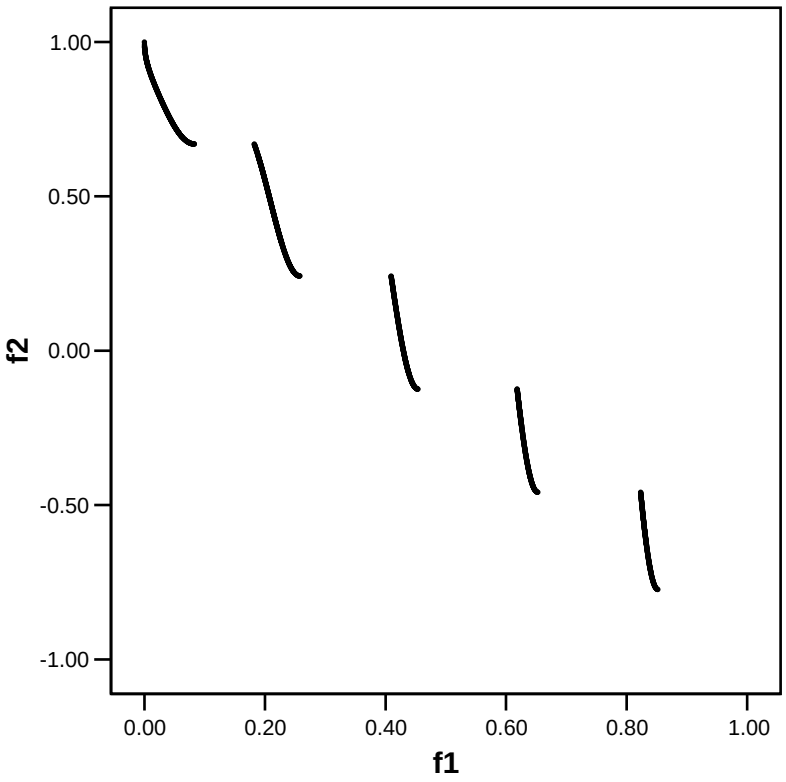


Figure 6: Pareto optimal front for ZDT3 function

3.1.1.4 ZDT4 Function

ZDT4 is a bi-objective MOP scalable in the number of variables. By Default, uses 10 real variables. ZDT4 (shown in Equation 3.5) has a non-convex and multimodal Pareto optimal front.

Definition:

$$
\begin{aligned}
& f_1(x) = x_1 \\
& f_2(x) = g(x)\left[1-\sqrt{x_i / g(x)}\right] \\
& g(x) = 1+10(n-1)+\sum_{i=2}^{n}\left[x_i^2 - 10\cos\left(4\pi x_i\right)\right]
\end{aligned}
\tag{3.5}
$$

where $n = 10$, $x_1 \in [0,1]$, $x_i \in [-5,5]$ and $i = 2,....,n$

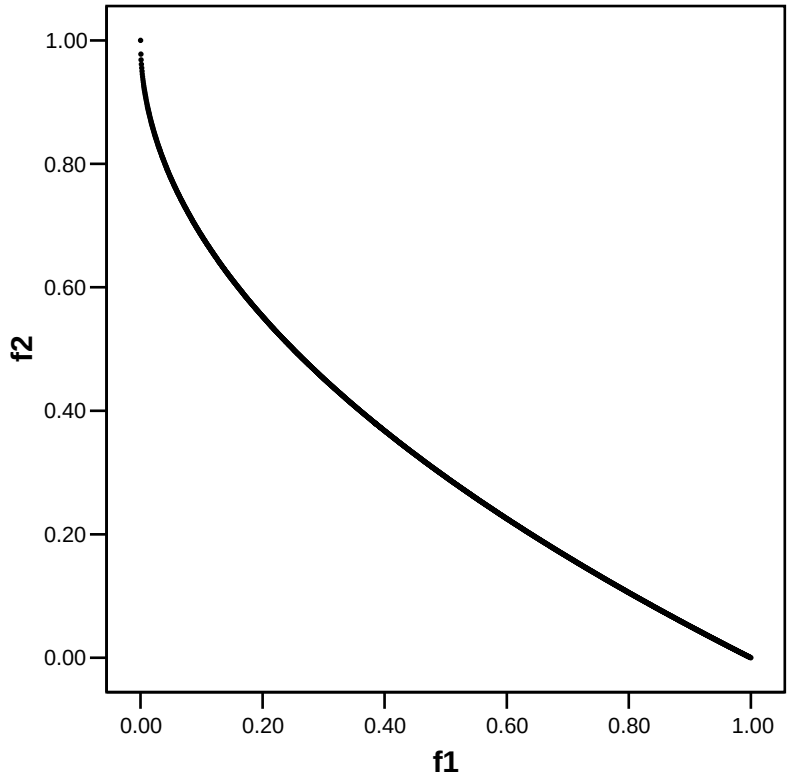


Figure 7: Pareto optimal front for ZDT4 function

3.1.1.5 ZDT6 Function

ZDT6 is a bi-objective scalable problem in the number of variables. By default this problems uses 10 real variables. ZDT6 (shown in Equation 3.6) has a non-convex and non-uniformly spaced Pareto optimal front.

Definition:

$$
\begin{aligned}
f_1(x) &= 1-\exp\left(-4x_1\right)\sin^6\left(4\pi x_1\right) \\
f_2(x) &= g(x)\left[1-\left(f_1(x)/g(x)\right)^2\right] \\
g(x) &= 1+9\left[\left(\sum_{i=2}^{n} x_i\right)/\left(n-1\right)\right]^{0.25}
\end{aligned}
\tag{3.6}
$$

where $n = 10$ and $x_i \in [0,1]$

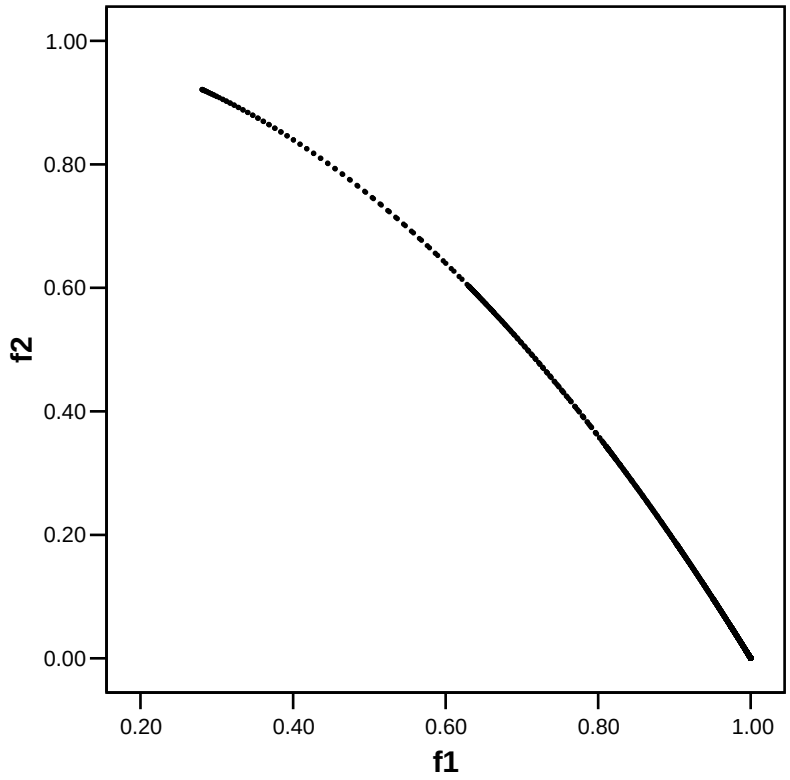


Figure 8: Pareto optimal front for ZDT6 function

3.1.2 DTLZ Test Functions

The DTLZ test functions introduced by Deb, Zitzler, Thiele and Laumanns [39], are a set of scalable problems with the ability to control difficulties in converging to the Pareto front and maintaining the diversity of solutions.

3.1.2.1 DTLZ1 Function

Here a DTLZ1 problem with 3 objectives and 7 variables is shown.

$$\begin{aligned}
&\text{minimise } f_1(x) = 1/2 x_1 x_2 ... x_{M-1}\left(1+g\left(x_M\right)\right),\\
&\text{minimise } f_2(x) = 1/2 x_1 x_2 ... \left(1-x_{M-1}\right)\left(1+g\left(x_M\right)\right),\\
&....\\
&\text{minimise } f_{M-1}(x) = 1/2 x_1 \left(1-x_2\right)\left(1+g\left(x_M\right)\right),\\
&\text{minimise } f_M(x) = 1/2\left(1-x_1\right)\left(1+g\left(x_M\right)\right).\\
&\text{subject to } 0 \le x_i \le 1, \textit{ for } \text{i=1,2,...,n.}
\end{aligned} \tag{3.7}$$

$$g\left(x_M\right) = 100\left[\left|X_M\right| + \sum_{x_i \in X_M}\left(x_i - 0.5\right)^2 - \cos\left(20\pi\left(x_i - 0.5\right)\right)\right]$$

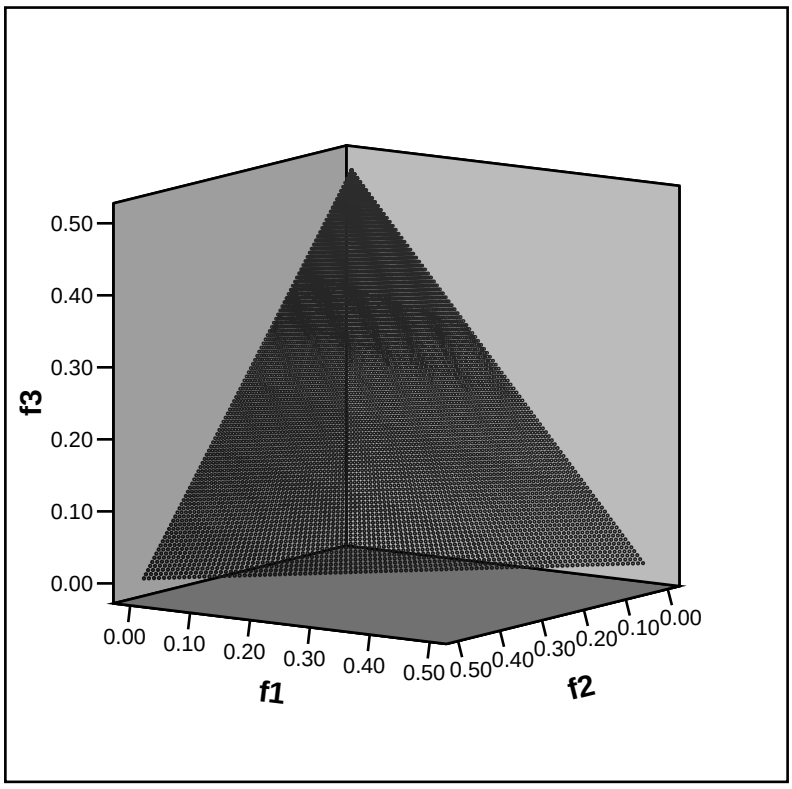


Figure 9: Pareto optimal front for DTLZ1 function

3.1.2.2 DTLZ2 Function

Here a DTLZ2 problem with 3 objectives and 12 variables is shown.

$$
\begin{aligned}
&\text{minimise } f_1(x) = (1+g(x_M))\cos(x_1\pi/2)\ldots\cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2),\\
&\text{minimise } f_2(x) = (1+g(x_M))\cos(x_1\pi/2)\ldots\cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2),\\
&\text{minimise } f_3(x) = (1+g(x_M))\cos(x_1\pi/2)\ldots\sin(x_{M-2}\pi/2),\\
&\ldots.\\
&\text{minimise } f_M(x) = (1+g(x_M))\sin(x_1\pi/2),\\
&\text{with } g(x_M) = \sum_{x_i}\in X_M (x_i - 0.5)^2,\\
&\textit{subject} \text{ to } 0 \le x_i \le 1, \ \textit{for} \text{ i=1,2,...,n.}
\end{aligned}
\tag{3.8}
$$

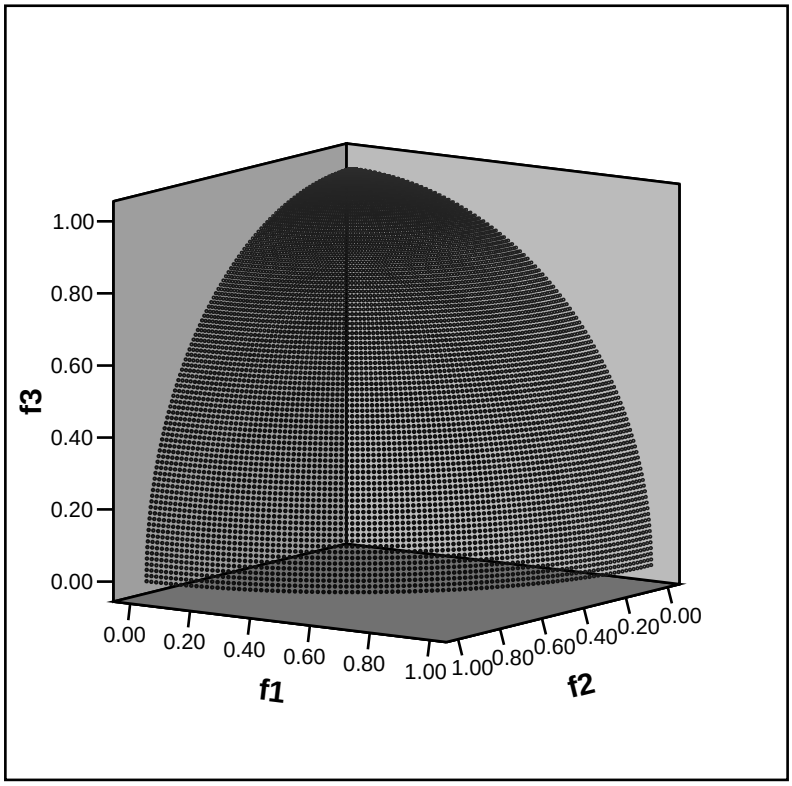


Figure 10: Pareto optimal front for DTLZ2 function

3.1.2.3 DTLZ3 Function

Here a DTLZ3 problem with 3 objectives and 12 variables is shown.

$$
\begin{aligned}
&\text{minimise } f_1(x) = (1+g(x_M))\cos(x_1\pi/2)\ldots\cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2),\\
&\text{minimise } f_2(x) = (1+g(x_M))\cos(x_1\pi/2)\ldots\cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2),\\
&\text{minimise } f_3(x) = (1+g(x_M))\cos(x_1\pi/2)\ldots\sin(x_{M-2}\pi/2),\\
&\ldots \qquad (3.9)\\
&\text{minimise } f_M(x) = (1+g(x_M))\sin(x_1\pi/2),\\
&\text{with } g(x_M) = 100\left[|X_M| + \sum_{x_i \in X_M}(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right]\\
&\textit{subject} \text{ to } 0 \le x_i \le 1, \ \textit{for} \text{ i=1,2,...,n.}
\end{aligned}
$$

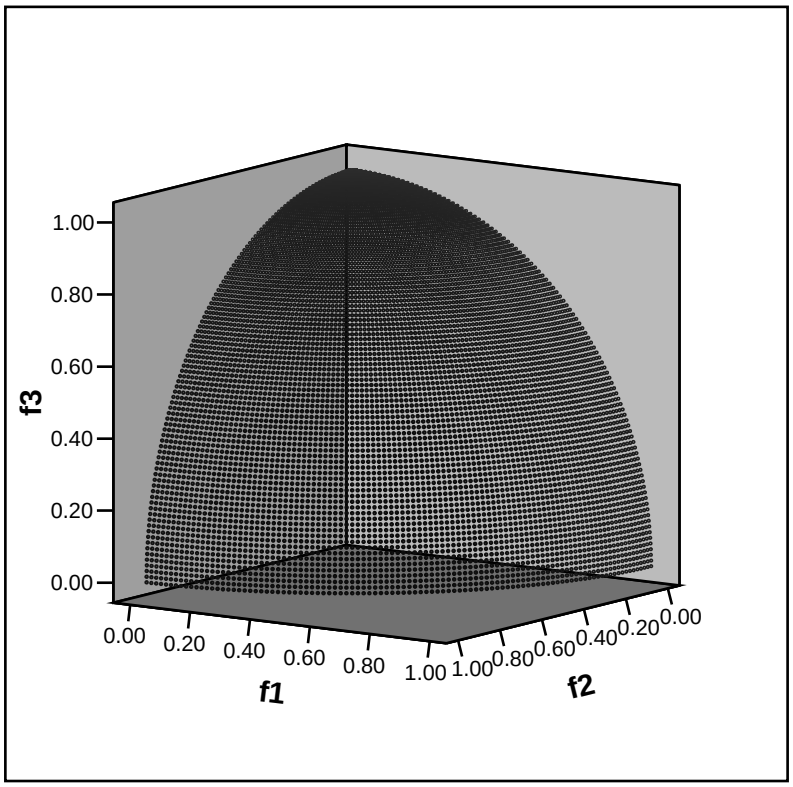


Figure 11: Pareto optimal front for DTLZ3 function

3.1.2.4 DTLZ4 Function

Here a DTLZ4 problem with 3 objectives and 12 variables is shown.

$$\begin{aligned}
&\text{minimise } f_1(x) = (1+g(x_M))\cos(x_1^{\alpha}\pi/2)\ldots\cos(x_{M-2}^{\alpha}\pi/2)\cos(x_{M-1}^{\alpha}\pi/2),\\
&\text{minimise } f_2(x) = (1+g(x_M))\cos(x_1^{\alpha}\pi/2)\ldots\cos(x_{M-2}^{\alpha}\pi/2)\sin(x_{M-1}^{\alpha}\pi/2),\\
&\text{minimise } f_3(x) = (1+g(x_M))\cos(x_1^{\alpha}\pi/2)\ldots\sin(x_{M-2}^{\alpha}\pi/2),\\
&\ldots\\
&\text{minimise } f_M(x) = (1+g(x_M))\sin(x_1^{\alpha}\pi/2),\\
&\text{with } g(x_M) = \sum_{x_i \in X_M}(x_i - 0.5)^2,\\
&\textit{subject to } 0 \le x_i \le 1, \ \textit{for } \text{i=1,2,...,n.}
\end{aligned} \tag{3.10}$$

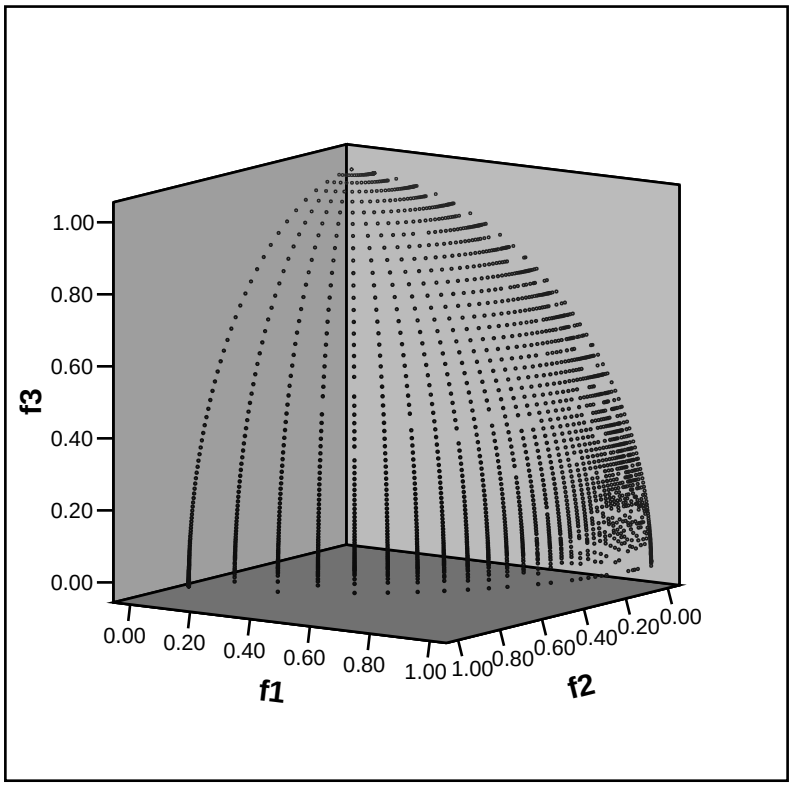


Figure 12: Pareto optimal front for DTLZ4 function

3.1.2.5 DTLZ5 Function

Here a DTLZ5 problem with 3 objectives and 12 variables is shown.

$$
\begin{aligned}
&\text{minimise } f_1(x) = (1+g(x_M))\cos(\theta_1\pi/2)\ldots\cos(\theta_{M-2}\pi/2)\cos(\theta_{M-1}\pi/2),\\
&\text{minimise } f_2(x) = (1+g(x_M))\cos(\theta_1\pi/2)\ldots\cos(\theta_{M-2}\pi/2)\sin(\theta_{M-1}\pi/2),\\
&\text{minimise } f_3(x) = (1+g(x_M))\cos(\theta_1\pi/2)\ldots\sin(\theta_{M-2}\pi/2),\\
&\ldots\\
&\text{minimise } f_M(x) = (1+g(x_M))\sin(\theta_1\pi/2),\\
&\text{with } g(\theta_i) = \pi/4(1+g(X_M)) + (1+2g(X_M)x_i) \text{ for } i = 2,\ldots,M-1,\\
&\text{with } g(x_M) = \sum_{x_i \in X_M}(x_i - 0.5)^2,\\
&\textit{subject} \text{ to } 0 \le \theta_i \le 1, \textit{ for } \text{i=1,2,...,n.}
\end{aligned}
\tag{3.11}
$$

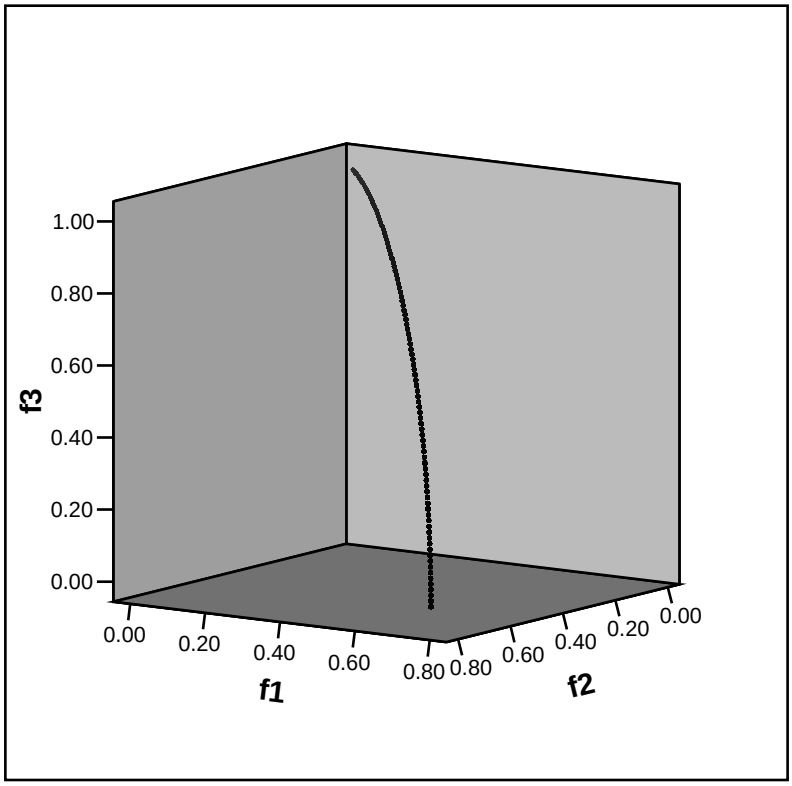


Figure 13: Pareto optimal front for DTLZ5 function

3.1.2.6 DTLZ6 Function

Here a DTLZ6 problem with 3 objectives and 12 variables is shown.

$$
\begin{aligned}
&\text{minimise } f_1(x) = (1+g(x_M))\cos(\theta_1\pi/2)\ldots\cos(\theta_{M-2}\pi/2)\cos(\theta_{M-1}\pi/2),\\
&\text{minimise } f_2(x) = (1+g(x_M))\cos(\theta_1\pi/2)\ldots\cos(\theta_{M-2}\pi/2)\sin(\theta_{M-1}\pi/2),\\
&\text{minimise } f_3(x) = (1+g(x_M))\cos(\theta_1\pi/2)\ldots\sin(\theta_{M-2}\pi/2),\\
&\ldots\\
&\text{minimise } f_M(x) = (1+g(x_M))\sin(\theta_1\pi/2),\\
&\text{with } g(\theta_i) = \pi/4(1+g(X_M)) + (1+2g(X_M)x_i) \text{ for } i = 2,\ldots,M-1,\\
&\text{with } g(x_M) = \sum_{x_i \in X_M} x_i^{0.1},\\
&\textit{subject to } 0 \le \theta_i \le 1, \textit{ for } \text{i=1,2,...,n.}
\end{aligned}
\tag{3.12}
$$

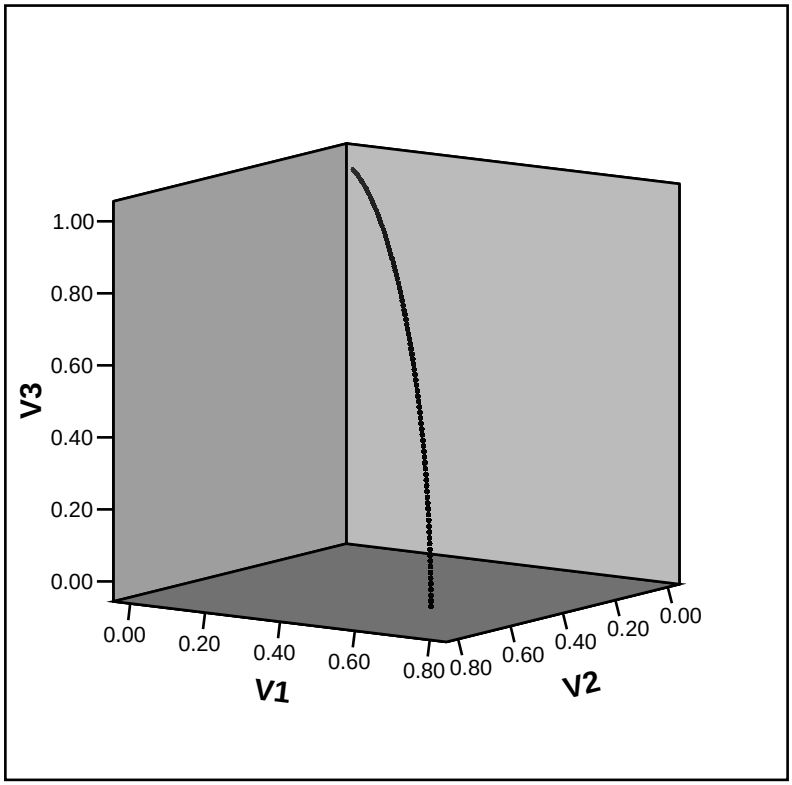


Figure 14: Pareto optimal front for DTLZ6 function

3.1.2.7 DTLZ7 Function

Here a DTLZ7 problem with 3 objectives and 22 variables is shown.

$$
\begin{aligned}
&\text{minimise } f_1(x) = x_1, \\
&\text{minimise } f_2(x) = x_2, \\
&\ldots. \\
&\text{minimise } f_{M-1}(x) = x_{M-1}, \\
&\text{minimise } f_M(x) = \left(1 + g(X_M)\, h(f1, f1, \ldots, f_{M-1}, g)\right). \\
&g(x_M) = 1 + \left(9 / |X_M|\right) \sum_{x_i \in X_M} x_i, \\
&h(f1, f1, \ldots, f_{M-1}, g) = M - \sum i = 1M - 1\left[\left(f_i / (1+g)\right)\left(1 + \sin(3\pi f_i)\right)\right], \\
&\text{subject to } 0 \le \theta_i \le 1, \textit{ for } \text{i=1,2,...,n.}
\end{aligned}
\tag{3.13}
$$

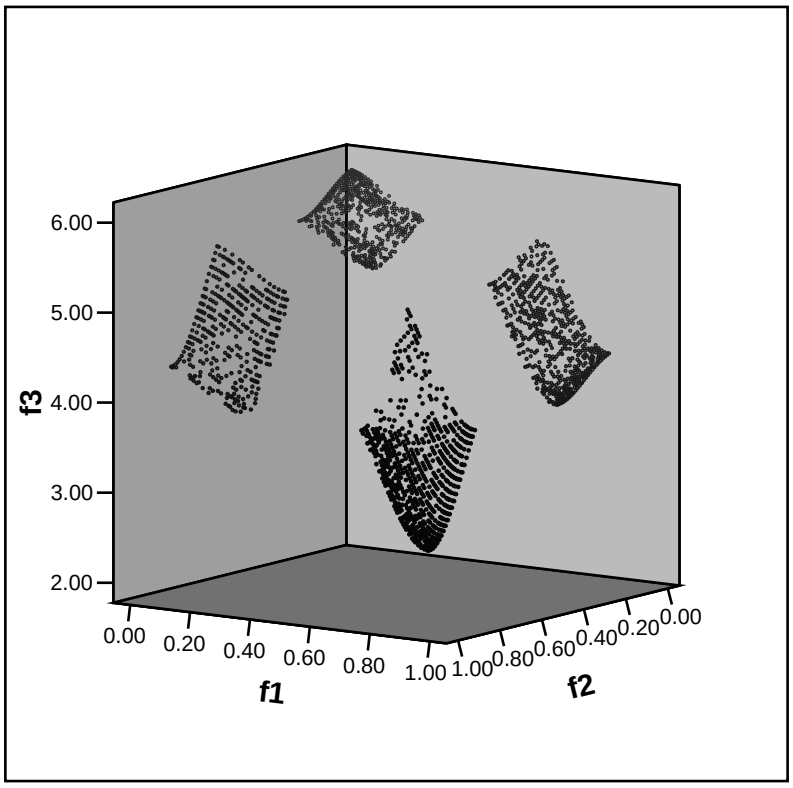


Figure 15: Pareto optimal front for DTLZ7 function

3.1.3 Schaffer Test Function (SCH)

Schaffer [40] test function is a bi-objective problem with one real variable. SCH (shown in Equation 3.14) has a convex and connected Pareto optimal front.

Definition:

$$f_1(x) = x^2$$
$$f_2(x) = (x-2)^2 \quad (3.14)$$

where $x \in [0,2]$

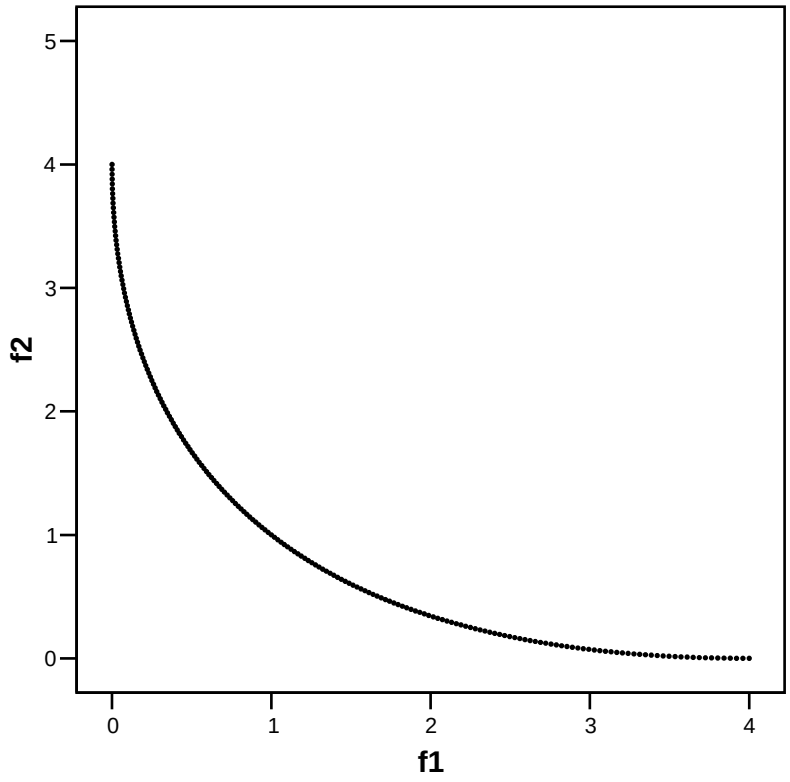


Figure 16: Pareto optimal front for SCH function

3.1.4 Kursawe Test Function (KUR)

Kursawe [41] test function is a bi-objective problem. KUR (shown in Equation 3.15) has a disconnected Pareto optimal front.

Definition:

$$f_1(x) = \sum_{i=1}^{n-1}(-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2}))$$
$$f_2(x) = \sum_{i=1}^{n}(|x_i|^{0.8} + 5\sin x_i^3) \quad (3.15)$$

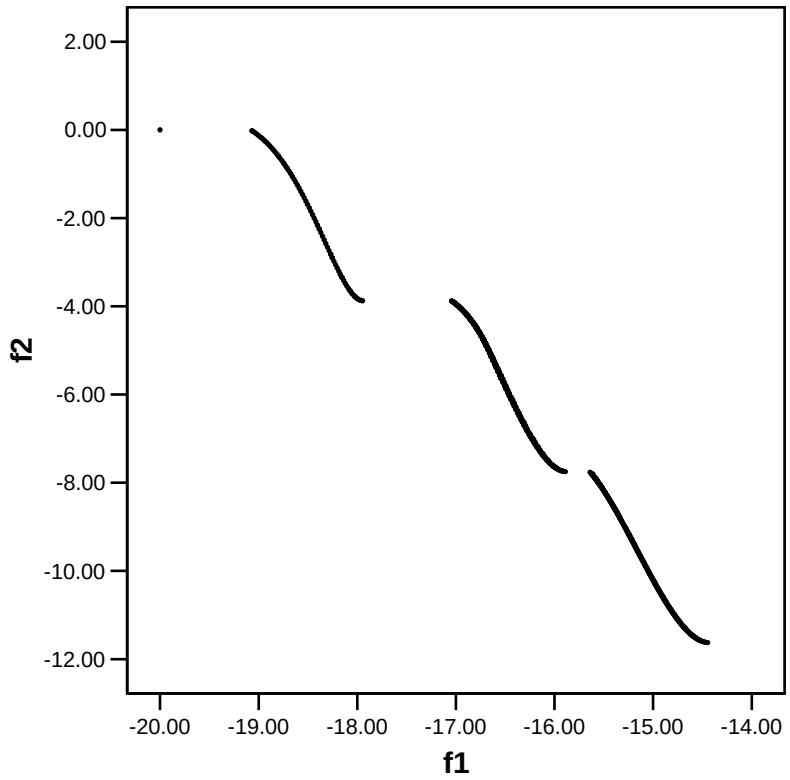


Figure 17: Pareto optimal front for KUR function

3.1.5 Fonseca Test Function (FON)

Fonseca [23] test function is a bi-objective problem with one real variable. FON (shown in Equation 3.16) has a non-convex and connected Pareto optimal front.

Definition:

$$f_1(x) = 1 - \exp(-\sum_{i=1} 3(x_i - \frac{1}{\sqrt{3}})^2)$$

$$f_2(x) = 1 - \exp(-\sum_{i=1} 3(x_i + \frac{1}{\sqrt{3}})^2)$$

(3.16)

where $x_1, x_2, x_3 \in \left[\frac{-1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right]$

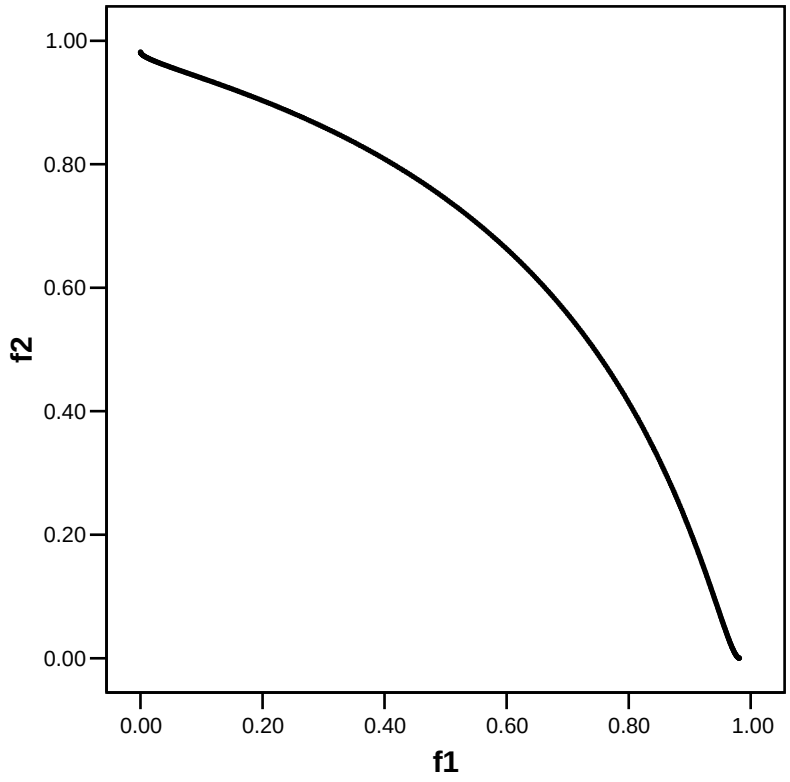


Figure 18: Pareto optimal front for FON function

3.2 Performance Metrics

Owing to the difficulty in defining a single and precise method of evaluating algorithm performance, it is essential to define several criteria evaluating procedures for doing the same. This task of performing accurate performance comparison between different algorithms is achieved by the use of several metrics defined in literature. Each of these metrics measure one or more particular aspects of an algorithm's performance. Generally the different aspects of interest when doing comparison are distance to the Pareto optimal front, the number of non-dominated elements in the obtained set, the spread of solutions, the quality of the non-dominated solution set etc. In this context, some of the important performance metrics are briefly reviewed.

3.2.1 Error Ratio (ER)

This metric proposed by Veldhuizen [37] is used to measure the ratio of those vectors that are in the true Pareto front to those which are not in the true Pareto front (PF_{known}). Therefore this metric uses the true Pareto front as a reference set. It is given as,

$$\frac{\sum_{i=1}^{n} e_i}{n} \tag{3.17}$$

where n is the number if vectors in the approximation set. e_i = 0 if vector i is in the true Pareto front and 1 otherwise. The lower the value of ER, the better the non dominated set will be.

3.2.2 S metric (S)

The S metric introduced by Zitzler and Thiele [28] is used to measure the size of the dominated space. This scaling independent metric gives a measure of the volume of the objective space that is weakly dominated by the non dominated set (A). As stated in [2] the S metric allows assessing the set of vectors independently.

3.2.3 Coverage of two sets (C)

This metric by Zitzler and Thiele [28] measures if one of the two sets of vectors is weakly dominated by the other.

$$C(A,B) = \frac{\left|\{b \in B / \exists a \in A : a \succeq b\}\right|}{|B|} \tag{3.18}$$

Let A, B ☐ X be two sets of decision vectors. The function C *maps the ordered pair* (*A*, *B*) *to the interval* [0, 1]*:*

The value C (*A*, *B*) = 1 means that all decision vectors in *B* are weakly dominated by *A*. The opposite, C (*A*, *B*) = 0, means that none of the points in *B* are weakly dominated by *A*.

3.2.4 Coverage Difference (D)

The D metric [2] was introduced to tackle certain anomalies that could occur in the C metric. It measures the coverage difference of two sets of decision vectors. It finds the volume space that is weakly dominated by one set but not by the other set of vectors.

Let A, B ☐ X be two sets of decision vectors.

The function D *is defined by*

$$D(A,B) = S(A+B) - (B) \tag{3.19}$$

and gives the size of the space weakly dominated by A but not weakly dominated by B (regarding the objective space).

3.2.5 Generational Distance (GD)

The Generational distance (Veldhuizen [37]), is the average distance from the obtained set to the true Pareto front. This metric also uses the true Pareto front as a reference set.

$$\frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{3.20}$$

where d_i is the Euclidean distance in the objective space from solution *i* to the nearest solution in the true Pareto front.

3.2.6 Maximum Pareto Front Error (MPFE)

This metric (Veldhuizen [37]) is used to measure the largest distance between any element in the obtained set and the respective closest element in the true Pareto front.

$$\max_j(\min_i \left| f_1^i(\vec{x}) - f_1^j(\vec{x}) \right|^p + \left| f_2^i(\vec{x}) - f_2^j(\vec{x}) \right|^p)^{1/p}$$

(3.21)

where $i = 1,\ldots..,n_1$ and $j = 1,\ldots.,n_2$ are elements in the obtained set and the true Pareto front respectively, and $p = 2$. The smaller the MPFE the better the solutions are.

3.2.7 Overall Non Dominated Vector Generation (ONVG)

This metric (Veldhuizen [37]) measures the size of the obtained non dominated set.

$$ONVG = \left| PF_{known} \right| \tag{3.22}$$

where PF_{known} is the approximation set.

3.2.8 Overall Non Dominated Vector Generation Ratio (ONVGR)

This metric (Veldhuizen [37]) measures the ratio of the size of the obtained set to the size of the true Pareto front.

$$ONVG = \frac{\left| PF_{known} \right|}{\left| PF_{true} \right|} \tag{3.23}$$

.

3.2.9 Spacing Metric (Deb)

This metric proposed by (Deb et al. [5]) is used to measure the diversity of the solution in the objective space by calculating the evenness of points in the obtained set.

$$\Delta = \sum_{i=1}^{|PF_{known}|} \frac{\left| d_i - \overline{d} \right|}{\left| PF_{known} \right|} \tag{3.24}$$

where d_i is the Euclidean distance between two consecutive elements in the non dominated front, and $\overline{d}$ is the average of these distances.

3.2.10 Spacing Metric (Schott)

This metric proposed by (Schott [38]) also measures the diversity of solutions in the front

$$S = \sqrt{\frac{1}{n-1} \sum_{i+1}^{n} \left(\overline{d} - d_i \right)^2}$$
$$where, d_i = \min_j \left(\left| f_1^i - f_1^j \right| + \left| f_2^i - f_2^j \right| \right) \tag{3.25}$$

$\overline{d}$ is the mean of all d_i and n = PF_{known}.

3.3 Summary

In this chapter the features and definitions of most widely used test problems and performance metrics are discussed. The graphs depicting the Pareto optimal fronts for the test problems are also shown for reference to the experimental study in chapter 5.

Chapter 4

Multi-objective Optimization Problems Toolbox (M-OPT)

M-OPT is a software application toolbox intended to demonstrate and solve multi-objective optimization problems (MOPs) using several popular nature inspired algorithms with the main focus on evolutionary multi-objective algorithms.

The central aim of this software package is to provide an environment which could be used not only for testing the performance efficiency of the current algorithms, but also to aid implementation of new methods by reusing the code. This is all the more easily done owing to the fact the package is developed in Java, thereby providing an object oriented approach to program coding in addition to its proven efficiency and portability. There are sets of base classes which form the building blocks and all the algorithm implementations are coded based on these base classes (for ex., classes for density estimation, genetic operators, test functions etc.) , thus making the comparison of the methods more authentic.

The implemented algorithms include the Non Dominated Sorted Genetic Algorithm (NSGA II) [5], Strength Pareto Evolutionary Algorithm (SPEA 2) [6], Pareto Archived Strategy (PAES) [31], Pareto Envelope Based Selection (PESA II) [30], Multi-objective Particle Swarm Optimization (OMOPSO) [47] and two new hybridized algorithms NSGAII* and SPEA2* (These are discussed in chapter 5).

Additionally most of the popularly used standard test functions used by researchers are implemented, which includes the five ZDT (Zitzler Deb Thiele [34]) functions, seven DTLZ (Deb Thiele Laumanns Zitzler [39]) functions and other unconstrained test problems such as Kursawe, Schaffer and Fonseca. Refer to section 3.1 for details on these test functions. The framework also provides a graphical user interface with real time plotting of the objective functions and variables. It also allows the user to modify the algorithm specific parameters thereby enhancing the scope of hybridization techniques.

4.1 General Architecture and Description

The software is developed in NetBeans IDE version 5.5 and JDK version 1.5. As mentioned before, each algorithm accesses the methods and variables of base classes, which defines the basic structure of the algorithm.

The class ***BaseStructure*** implements a generic template for the algorithms developed in M-OPT. Every algorithm must have a mapping between the parameters and their names, and another mapping between the operators and their names. This class declares an abstract method called *run()*, which defines the behavior of the algorithm. This class has inherited by all the algorithms and it provides methods like *paramGet()* and *paramAdd()* to access parameter required for the application. The genetic operators used in the algorithms are provided by *operAdd()* and *operGet()* methods. The *run()* method starts the execution of the algorithm. The class ***Population*** represents the set of solution objects which in turn are composed of the ***chromosome*** object which again contains the ***gene*** component, following the evolutionary algorithm terminology. The *gene* class is an interface defining the array of variables having different representation namely binary, real or real-coded binary. The class ***Problem*** defines the problem that the algorithm solves. It uses the *assess()* method to assess the problem function. The genetic operators like crossover, mutation and selection are defined in the ***Operator*** class. Other classes are also defined to find the ranking, crowding distance, density etc.

4.2 Discussion: NSGA-II

Here the implementation of NSGAII is discussed briefly to give a better picture of the working of the algorithm in M-OPT. The algorithm specific parameter values are accepted from the GUI framework and configured using the *NSGA_init()* method. The ***NSGAII*** class defines the execution of the algorithm and specifies the problem to solve, the operators to use etc. The NSGA_init() method is shown in Fig.1 and an extract from the NSGAII class is included as Fig.2.

The input parameters such as the population and the maximum number of evaluations to compute are accepted in the line 9 to 13 and the problem is selected in line 14. In line15, a new instance of the NSGAII class is created. Lines 16 and 17 are used to set the parameters for the algorithm. Next (lines 18 to 25), crossover, mutation, and selection operators are specified. After this the operator are assigned to the algorithm using the operAdd() method. The line 30 starts the execution of the

algorithm with lines 29 and 31 used to determine the execution time in milliseconds. After this, the objective function values and variables are written into text files, in lines 36 and 37.

```
1.  ProblemClass probObj = null   ;
2.  BaseStructure algorithm ;        // The algorithm to be used
3.  OperatorClass  crossover ;        // Crossover operator
4.  OperatorClass  mutation  ;        // Mutation operator
5.  OperatorClass  selection ;       // Selection operator

6.  int popSize, maxEvaluations;
7.  double probCross, crossoverDist, mutationDist;

8.  try{

9.  popSize       = Integer.parseInt(popSizeTF.getText());
10. maxEvaluations       = Integer.parseInt(maxEvaluationsTF.getText());
11. probCross     = Double.parseDouble(probCrossTF.getText());
12. crossoverDist = Double.parseDouble(crossoverDistTF.getText());
13. mutationDist  = Double.parseDouble(mutationDistTF.getText());

14. probObj = initProb(ProblemCB, RepresentationCB, probObj);

15. algoObj = new NsgaII(probObj);

16. algoObj.initializeInput("populationSize",popSize);
17. algoObj.initializeInput("maxEvaluationsuations", maxEvaluations);

18. crossVar =
    CrossoverClass.opCrossoverGet(crossoverCB.getSelectedItem().toString());
19. crossVar.paramSet("probability",probCross);
20. crossVar.paramSet("distribuitionIndex",crossoverDist);

21. mutation =
    MutationClass.getMutationOperator(mutationCB.getSelectedItem().toString());
22. mutation.paramSet("probability",1.0/probObj.getNumberOfVariables());
23. mutation.paramSet("distributionIndex",mutationDist);
24. .paramSet("probability",1.0/80);

25. selection = new BinTournament();

26. algoObj.operAdd("crossover",crossVar);
27. algoObj.operAdd("mutation",mutation);
28. algoObj.operAdd("selection",selection);

29. long startTime = System.currentTimeMillis();
30. Population population = algoObj.run();
31. long timeOfExecution = System.currentTimeMillis() - startTime;
32. System.out.println("Total time of execution: "+timeOfExecution);
33. JOptionPane.showMessageDialog(null,"Run Successful", "Status",
    JOptionPane.INFORMATION_MESSAGE);
```

```
34. execLbl.setText(timeOfExecution+" ms");
35. fileLabel1.setText("Objectives values have been written to file
    Obj_NSGAII.txt");

36. population.setOutputObjectives("Obj_NSGAII.txt");
37. population.setOutputvariables("Var_NSGAII.txt");
38. fileLabel2.setText("Variables values have been written to file
    Var_NSGAII.txt");
39.
40. }catch(Exception e){
41. JOptionPane.showMessageDialog(null,"Please Fill Up All The Fields ", "Data
    Validation",    JOptionPane.INFORMATION_MESSAGE);
42. }
```

Figure 19: Initialization of NSGAII - NSGA_init () method.

A piece of code of the class NSGAII is shown in Figure. 2. The execution of the algorithm is performed in the run() method in line 11. Initially the parameter values and operators are obtained. Then, the two populations required by the algorithm are population is initialized. The main loop of the algorithm starts in line 13. It follows the genetic algorithms steps: two parents are selected (lines 19-20), a pair of children is obtained after crossover (line 22), the mutation (lines 23-24). Then they are evaluated (lines 25-28), and finally inserted into the child population (lines 29-30).

```
1.  package EA_TBox.MoeaAlgorithms;
2.  import EA_TBox.base.*;
3.  import EA_TBox.base.operator.comparator.DominanceComparator;
4.  import EA_TBox.base.BaseStructure;
5.  import java.util.Comparator;
6.  import EA_TBox.util.*;

7.  public class NsgaII extends BaseStructure {

8.  public NsgaII(ProblemClass probObj){
9.  this.probObj_ = probObj;
10. }

11. public Population run() {
12. ….// initialize input variables and operators

13. while (noOfGenerations< maxEvaluations) {

14. //-> Create the child population
15. childPop  = new Population(populationSize);
16. Individual [] parents = new Individual[2];
17. for (int i = 0; i < (populationSize/2); i++){
```

```
18. //obtain parents
19. parents[0] = (Individual)selectionOperator.run(population);
20. parents[1] = (Individual)selectionOperator.run(population);
21. if (noOfGenerations< maxEvaluations) {
22. Individual [] childMembers = (Individual []) crossoverOperator.run(parents);
23. mutationOperator.run(childMembers[0]);
24. mutationOperator.run(childMembers[1]);
25. probObj_.assess(childMembers[0]);
26. probObj_.assessConstraints(childMembers[0]);
27. probObj_.assess(childMembers[1]);
28. probObj_.assessConstraints(childMembers[1]);
29. childPop.add(childMembers[0]);
30. childPop.add(childMembers[1]);
31. noOfGenerations+= 2;
32. } else {
33. childPop.add(new Individual(parents[0]));
34. childPop.add(new Individual(parents[1]));
35. } // if
36. } // for
37. //<-

38. //-> Create the population union of population and childMembers
39. union = ((Population)population).union(childPop);

40. ….// Ranking and crowding distance calculation phase
41. }
```

Figure 20: Components of program NsgaII.java

Next, the ranking and crowding estimation phase of NSGA-II: the two populations are joined and ranked. Then the newly formed population is obtained selecting the best ranked individuals, applying crowding distance to choose the best ones in the last selected rank. The plot obtained for NSGA II using M-OPT for the Schaffer test function is shown in Figure 22. A plot of the original Schaffer function is also included in Figure 23 along with screenshots of the toolbox in Appendix D. The toolbox also provides visualization of graphs through an interface to Matlab.

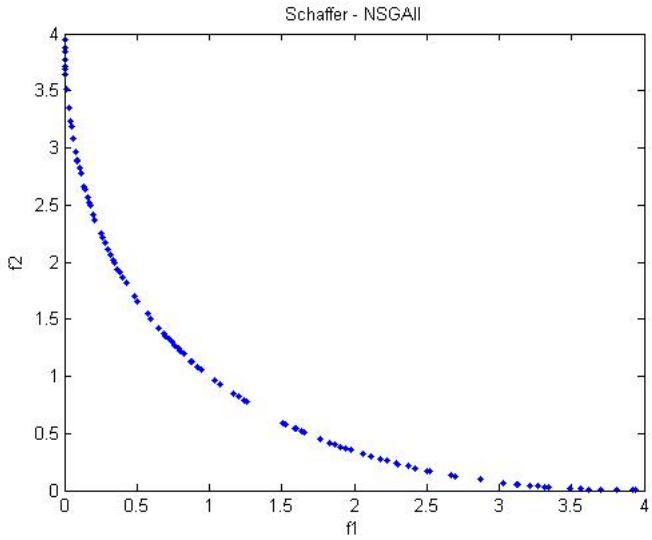


Figure 21: Schaffer Function M- OPT

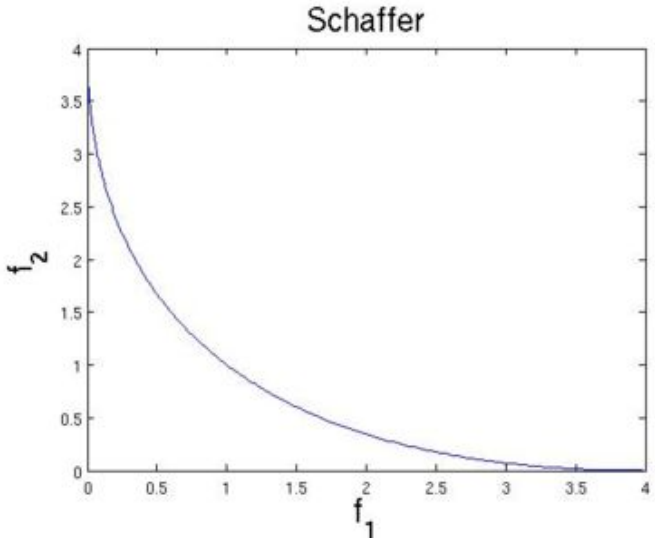


Figure 22: Original Schaffer Function

Chapter 5

Diversity: Analysis and Hybrid methods

5.1 Introduction

In this chapter, the diversity techniques used by some of the widely used MOEAs are discussed. Additionally, in section 5.3 modified algorithms are proposed for NSGA II and SPEA2 by introducing changes in the diversity estimation techniques of these methods.

5.2 Diversity preservation methods used in MOEAs

The efficiency of the output produced by multi-objective evolutionary algorithms depends heavily on an effective diversity preservation method. Diversity techniques simply put, enhances a wide spread of solutions along the objective space. Lack of an effective diversity method could result in the algorithm being driven towards a local Pareto, thereby missing out on the other potentially important areas of the search space. Usually diversity is achieved by manipulating density information pertaining to the individual in the search space. The spread of solutions is accomplished by excluding those solutions which are attributed with higher density values, from the selection process at each stage. Consequently, more and more diverse individuals are allowed to take part in the process of driving towards the true Pareto front. Although usage of diversity methods existed from the early part of MOEA research, its relevance became more appreciated later on whereby several methods were introduced. Goldberg introduced the Pareto Niching and Fitness sharing methods [20]. Improved methods were developed by Srinivas and Deb in their NSGAII [5] which uses crowding estimation technique for diversity maintenance. Knowles and Corne introduced the adaptive grid algorithm in PAES [33]. Equally popular is the archive truncation method using the nearest neighbor estimation technique developed by Zitzler et al. for their SPEA2 algorithm [6]. These current approaches in diversity techniques are briefly discussed in the following section.

5.2.1 Pareto Niching and Fitness Sharing

Pareto niching and fitness sharing methods have been used in single objective optimization for finding several optimum points in the search space. However, in MOEAs fitness sharing is performed with the goal of finding well distributed as well as well spread vectors. Fonseca and Fleming's [23] MOGA used fitness sharing in a restricted manner i.e. only those solutions which evaluate to vectors with identical Pareto rank can take part in fitness sharing. They measure niching distance in phenotypic space; the distance between two solutions' evaluated fitness vectors is computed and compared to σ_{share} (the key sharing parameter). If the distance is less than σ_{share}, the solution's associated niche count is then adjusted.

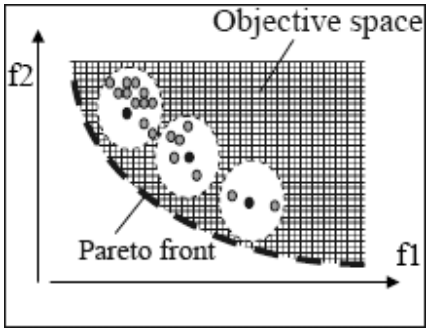


Figure 23: Pareto Niching

The NSGA differs slightly using a method where distance is measured in genotypic space; the distance between two solutions is compared to σ_{share}. The NSGA also shares fitness only between solutions evaluating to vectors with identical Pareto rank. Horn et al. [25] define niching differently in their Niched Pareto Genetic Algorithm (NPGA), which performs selection via binary Pareto domination tournaments. Solutions are selected if they dominate both the other and some small group (t_{dom}) of randomly selected solutions, but fitness sharing occurs only in the cases where both solutions are non dominated [24]. Each of the two solution's niche counts is derived by counting the number of objective vectors within σ_{share} of their evaluated vectors in phenotype space. The solution with a smaller niche count (fewer phenotypical neighbors) is then selected. This method was termed as *equivalence class sharing* by Horn et al.

The disadvantage of all these methods lies in the requirement of setting the key sharing parameter which is a crucial element of this method. Additionally the size of the population also affects the performance of fitness sharing method. Assigning

appropriate values to σ_{share} is generally difficult as it usually requires some *a priori* knowledge about the shape and separation of a given problem's niches, as stated in [24]. These disadvantages forced researchers to investigate a method which doesn't require the explicit knowledge of a sharing function value.

5.2.2 The Archive Truncation method

This diversity preservation method is used in SPEA2. Here, unlike NSGA-II, the selection process and density estimation go hand in hand with the fitness assignment. The algorithm considers two factors when it comes to finding non-domination, for every individual the number of individuals dominated by and the number which it dominates are determined. This way individuals dominated by the same members of the archive will have different fitness values [6]. As the selection process ideally has to give preference to the solutions belonging to the non-dominated set in the combined population, a raw fitness value is assigned denoting the number of individuals each individual dominates. This is referred to as the strength value S_i and the raw fitness of an individual is the sum of all strength values of the individuals it dominates. A high raw fitness value shows that the individual is dominated by many other individuals. So a lower value is preferred for the raw fitness. The raw fitness is given as:

$$R(i) = \sum_{j \in P_t + \overline{P}j, j \succ i} S(j)$$

where $S(i) = \left| \{ j \mid j \in P_t + \overline{P}_t \Lambda i \succ j \} \right|$

"$|....|$" Denotes the cardinality of the set, + stands for multiset union and $\succ$ stands for the Pareto dominance relation.

Although the raw fitness itself is a good density measure, it would not give accurate results when most individuals do not dominate each other. Therefore the fitness value is improved by adding more information. This is done with the k^{th} nearest neighbor estimation technique according to which the density of any point is a decreasing function of the distance to the k^{th} nearest element [6]. Thus the inverse of the distance to the k^{th} nearest gives the density estimate. It is given as

$$D(i) = \frac{1}{\sigma i^{k} + 2}$$

where $k = \sqrt{N + \overline{N}}$ i.e. [6] the square root of the sample size.

This measure added with the raw fitness determines the final fitness value $F(i)$. It is given as follows:

$$F(i) = R(i) + D(i)$$

An external archive is used for storing the solutions in each generation. The size of the archive is specified in advance and the non dominated set of solutions is copied into the archive in each generation run. If the number of non dominated solutions found in each run is lesser than the maximum archive size then the next best solutions are populated into the archive. On the other hand if the number of non dominated solutions is greater than the archive size, then archive truncation method is executed. This method removes solutions from the overfilled archive, based on the kth distance found previously, such that the individual which has the least distance to the other individual is removed from the archive. This process is continued until the required archive size is achieved.

5.2.3 The Crowding Distance technique

This method is used in the NSGA II, discussed in section 2.3.3. The two offspring populations generated from the parent population are joined and non dominated sorting is performed to generate fronts. Then the new population is filled with solutions of different non-dominated fronts with the best front given preference. The other fronts are deleted. The niching method is required when there are not enough slots available for the solutions. So the crowding distance method is used to identify the most crowded region in the fronts. The sorting of solutions is done based on the objective function. Infinite values are assigned to the extreme solutions. Then the distance measure is updated based on the difference in objective values between neighboring solutions. Thus to find the density of solutions around a particular solution, the average distance of two other points on either side of the current point is estimated along the objective values [5]. In other words this is the size of the largest cuboid enclosing the point. This measure gives the crowding distance as mentioned in [5]. Figure 24, obtained from [5] illustrates this calculation, where the average side length of the cuboid gives the crowding distance.

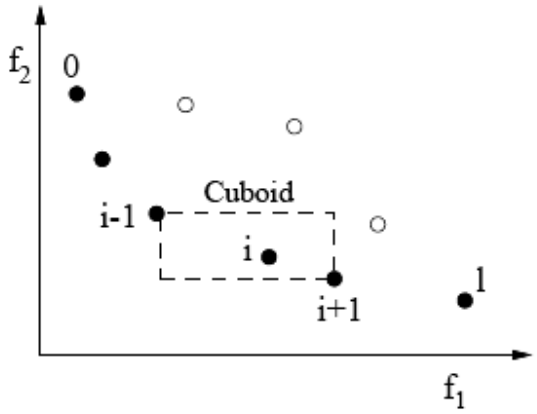


Figure 24: The crowding distance calculation (Source: Deb et. al [5])

This process is continued and finally each solution is assigned a distance value based on the objective function. Following this, the most crowded region is identified as the one with the least crowding distance value and consequently removed from further selection. This is achieved with the help of a crowded comparison operator, which eventually drives the population towards a well spread set of optimal solutions. As stated in [5], the crowded comparison operator ($\prec_n$) using the non-domination rank (i_{rank}) and the crowding distance (i_{dist}), is calculated with the following formula.

$$i \prec_n j \quad if\,(i_{rank} < j_{rank}) or ((i_{rank} = j_{rank}) and (i_{dist} > j_{dist})$$

Thus preference is always given to the solution with the better rank. Otherwise the solution in the least crowded region is preferred.

5.2.4 The Adaptive Grid Algorithm technique

Another method of density estimation is through histograms. These methods use a hyper grid to define neighborhoods within the objective space. The number of solutions present in the hyperboxes defines the density value. In Pareto archived evolution strategy (PAES) [33], the adaptive grid spacing is determined by a number of bisections specified by the user. These bisections of the objective range are defined by the locally non-dominated solutions. PAES also uses an external archive for storing non dominated individuals. In the case of the archive exceeding the preferred size, a solution from the most crowded hyperbox is selected for removal and is replaced by the newly found solution.

5.3 Improvements to Existing MOEAs

In this section two new hybrid MOEAs namely NSGAII* and SPEA2* are introduced. Two of the popular MOEAs, NSGAII and SPEA2 are modified with different diversity techniques replacing their original ones. NSGAII* uses the nearest neighbor estimate for measuring diversity while SPEA2* uses the crowding distance operation in its diversity mechanism.

5.3.1 NSGAII with Nearest Neighbor diversity (NSGAII*)

This hybrid algorithm is defined in the same manner as the original NSGAII with the difference in the density estimation procedure. NSGAII* uses the k-th nearest neighbor technique used by SPEA2, to determine the distance measure. The algorithm is implemented in the M-OPT toolbox discussed in chapter 4.

5.3.2 SPEA2 with Crowding Distance diversity (SPEA2*)

SPEA2* also maintains the same steps as the original SPEA differing only in the diversity mechanism. SPEA2* uses the crowding distance assignment technique to determine the distance measure. The algorithm is also implemented in the M-OPT toolbox.

Results of the performance analysis of the above mentioned algorithms, based on several test functions are listed in the chapter 6, followed by results discussion.

5.3.3 Summary

In this chapter different diversity measuring techniques used in MOEAs are discussed. Additionally, two new hybrid methods are proposed based on changes in the diversity calculation.

Chapter 6

Results and Discussion

6.1 Introduction

In this chapter the results of the experiments carried out in analyzing the performance of the algorithms relevant to this thesis, are tabulated and analyzed. The chapter is divided into sections representing the testing methodology used, the parameter setting for the algorithm runs, the performance metrics used, the summary of results in tables and the discussion.

6.2 Algorithms Runs

Each one of the algorithms is run with three different parameters for maximum number of evaluations with values being 5,000, 7,000 and 12,000. This way, four of the algorithms namely NSGAII, SPEA2, NSGAII*, SPEA2* are run for each of the different maximum number of evaluations. This process is repeated for each of the test problems namely Schaffer' test function (refer section 3.1.3), ZDT1 (refer section 3.1.1.1) and ZDT3 (refer section 3.1.1.3). Furthermore, the performance metrics for Spacing, Coverage, Generational distance, Coverage Difference, S metric, D metric and the time of execution (milli seconds) are also calculated in order to evaluate the performance of the algorithms in question. Each of the above mentioned process is iterated for 10 separate runs to avoid ambiguities through random results. After this the average, minimum, maximum and standard deviation values of those runs are also tabulated for comparison. Graphs representing the obtained Pareto front for the test problems are depicted too.

6.3 Testing environment

The M-OPT toolbox is used to execute the algorithms and generating the results. An open source software GUIMOO is used for generating the performance metrics [43].

6.4 Parameter Settings

The following parameters have been kept the same for all executions for fair comparison.

Population size	100
Crossover type	SBX
Distribution index for SBX	20
Crossover probability	0.9
Mutation type	Polynomial mutation
Distribution index for polynomial mutation	20

Table 2: Default parameter setting

6.5 Discussion on results

The diversity comparison between the four algorithms is shown in graphs below, represented by figures 26-28. The graphs are defined by average spacing metric measures plotted against the different variations of maximum generation numbers. Convergence is measured through Generational distance which is visualized in figures 29-31. The metrics spacing and the S metric are independent measures as mentioned in chapter 3. The other metrics such as generational distance, coverage, coverage difference and the D metric are measured against the respective true Pareto fronts.

The data is extracted from the results of averages, minimum, maximum and standard deviation values summarized from the actual runs. These summary results are shown in Tables 2-10. The value which ranked best for each metric is highlighted in the tables. For detailed results on the runs refer to tables in Appendix B. The non dominated front for each of the different algorithms on the test functions are visualized in section 6.6.

Based on the results obtained it has been found that the NSGAII* showed better diversity values (lower spacing metric measure) and better convergence (lower generational distance value) in the initial generations (5000) of the run for the Schaffer test problem. The SPEA2* outperformed others in diversity for the convex ZDT1 problem, where the even spread of solutions was shown to improve in the 12000 evaluation run (Figure 27. Again, for the non-convex disconnected problem ZDT3, SPEA2* produced the best results for the final generation runs (Figure 28).

The size of the non-dominated front measured by the S metric was used to find the overall quality of the non-dominated set. NSGAII and NSGAII* showed better results

for this metric in Schaffer's function, whereas SPEA2 performed well on ZDT1 and ZDT3 functions.

All the other three algorithms performed better than the SPEA2* in terms of convergence measures. NSGAII showed better results in generational distance measures denoting effective convergence, closely followed by NSGAII* and SPEA2.

6.5.1 Diversity analysis results

Figure 25: Spacing metric comparison for Schaffer function

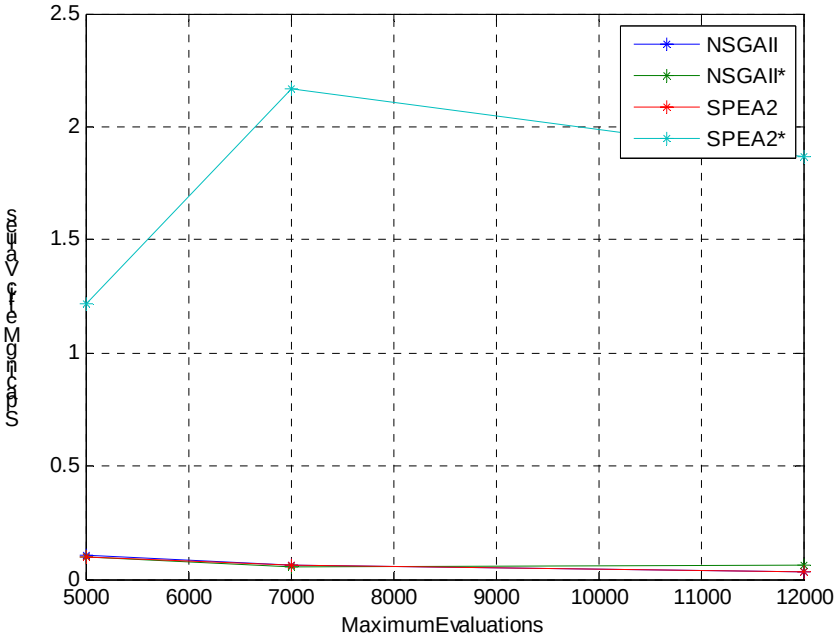


Figure 26: Spacing metric comparison for ZDT1 function

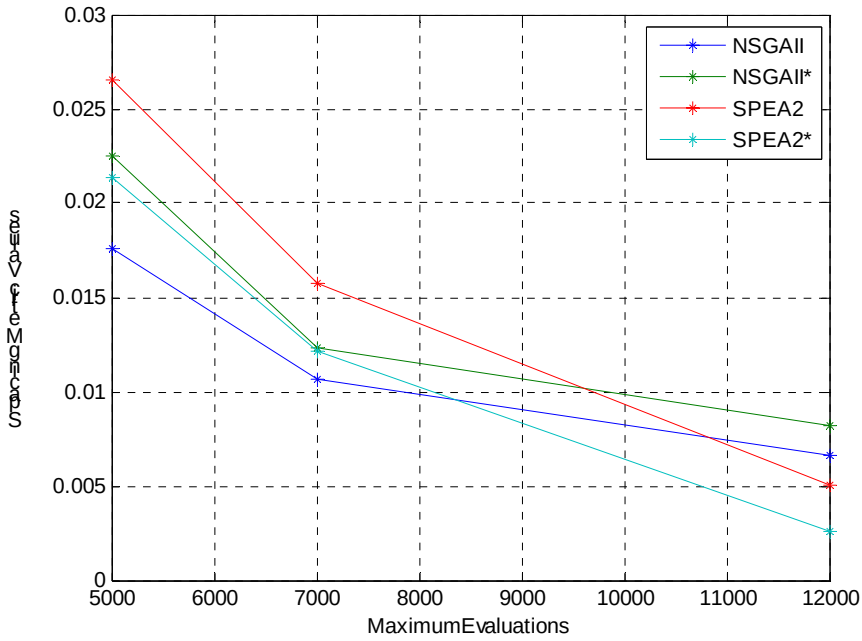
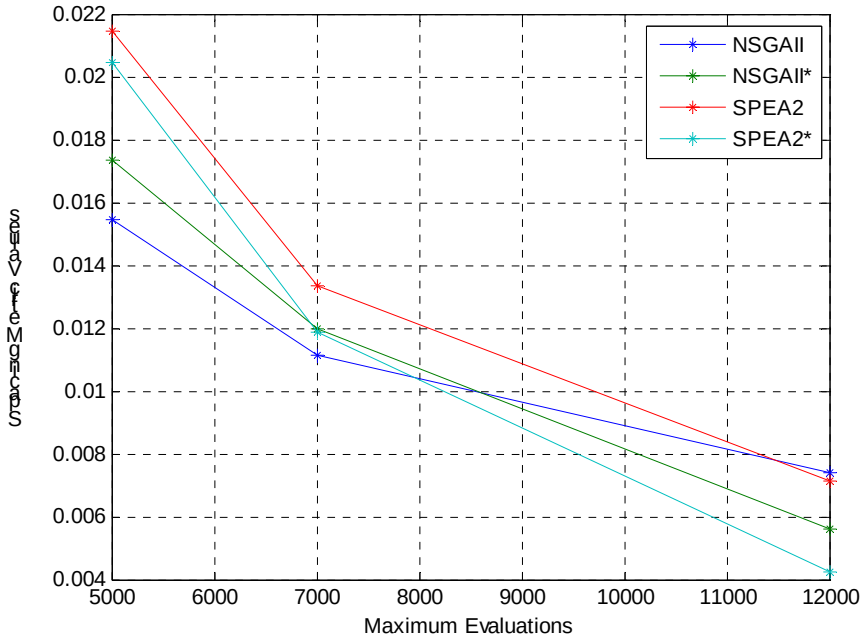


Figure 27: Spacing metric comparison for ZDT3 Function



6.5.2 Convergence analysis results

Figure 28: Generational Distance comparison for Schaffer Function

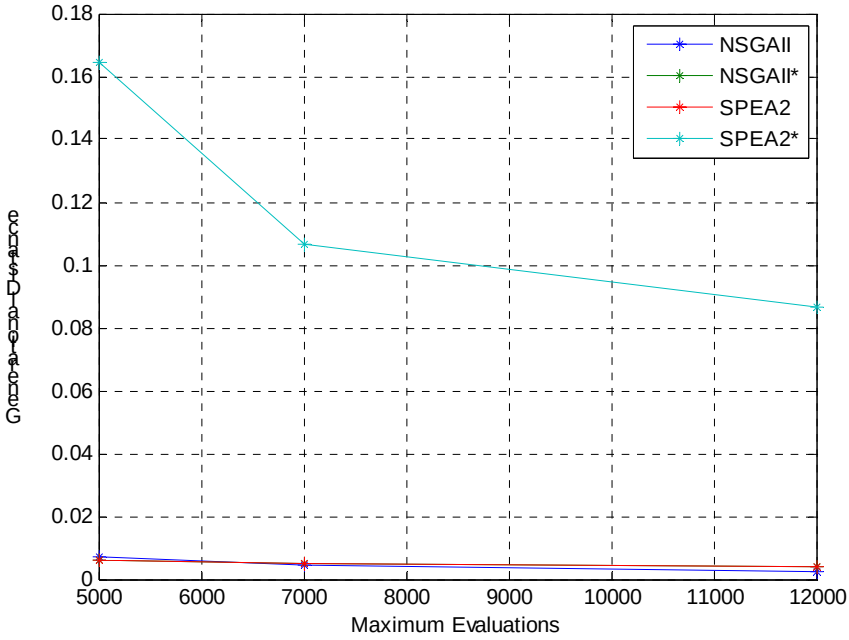


Figure 29: Generational Distance comparison for ZDT1 Function

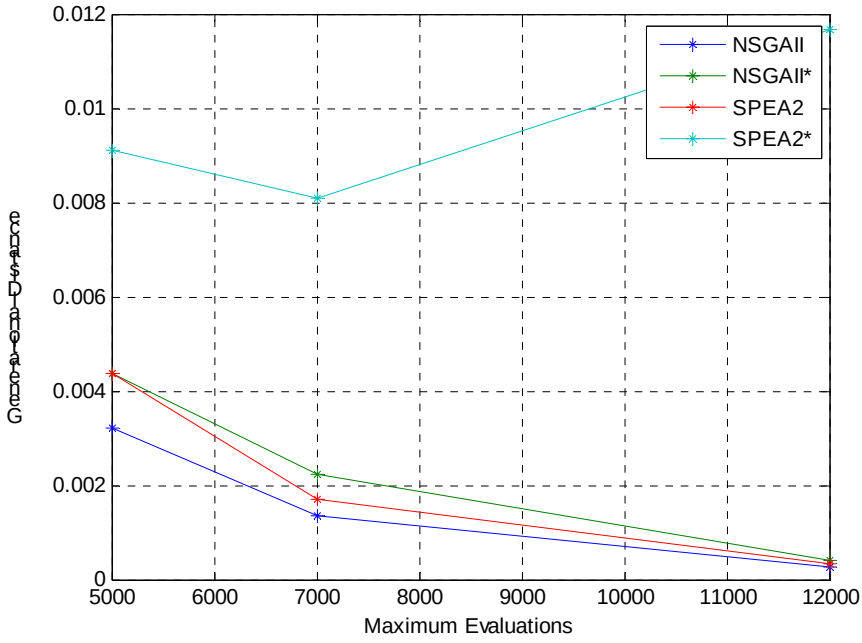
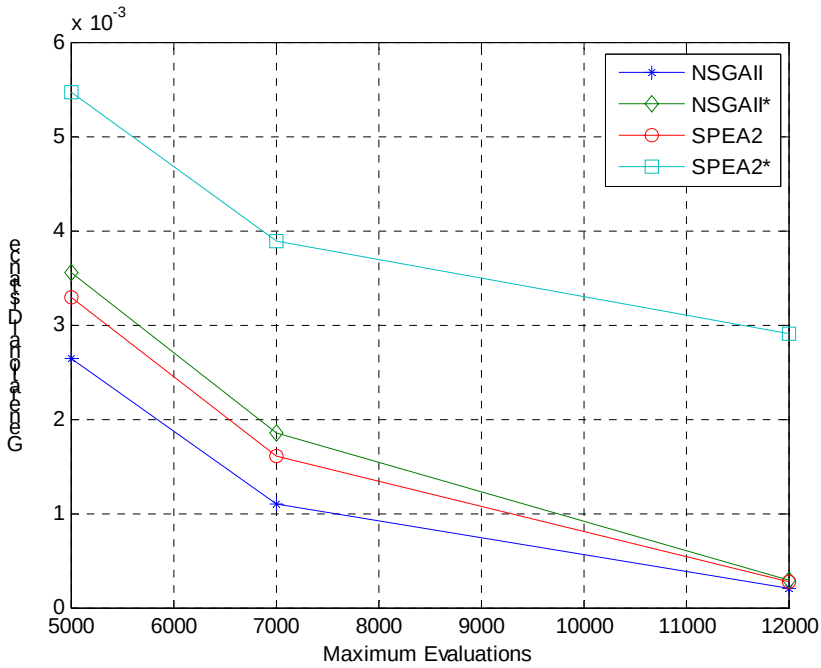


Figure 30: Generational Distance comparison for ZDT3 Function



6.5.3 Schaffer Test Function - Summary of results

Test Function : Schaffer								
Maximum Evaluations : 5000								
MOEA	**Metric**	**Spacing**	**S Metric**	**Generational Distance**	**Coverage**	**Coverage Difference**	**D Metric**	**Execution time**
NSGAII	Avg	0.106853	**14.81131**	0.006991	0.024998	-0.22868	0.002012	2151.6
	Min	0.049581	14.7231	0.00608	0	-0.79272	0.001501	2078
	Max	0.14227	14.903	0.007702	0.05	0.218391	0.003064	2250
	Sd	0.025791	0.05238	0.000542	0.013792	0.379793	0.000433	50.71313658
SPEA 2	Avg	0.10049298	14.24558	0.007042425	0.02773664	-0.22465415	0.001834645	**1965.4**
	Min	0.0744838	14.1668	0.00544966	0.0192308	-0.651938	0.00127214	1906
	Max	0.116793	14.2937	0.00911255	0.05	0.324505	0.00229137	2000
	Sd	0.012468918	0.039963867	0.001130666	0.01066551	0.317397945	0.000290473	31.79168305
NSGAII*	Avg	**0.09568323**	14.04266	**0.006303698**	0.02147422	0.05239508	0.002091587	10993.7
	Min	0.0680044	13.9572	0.00477164	0	-0.745261	0.00139909	9281
	Max	0.130139	14.1019	0.0076099	0.0465116	1.4067	0.00247652	12031
	Sd	0.023330897	0.043725131	0.001076192	0.017303366	0.633296465	0.000313343	1029.79567
SPEA2*	Avg	1.21908	61.05524	0.164619	0.380952	-3.06739	0.00028	4403.1
	Min	0	0	0.051227	0	-38.1212	0	4359
	Max	3.65446	79.6865	0.495977	1	24.2354	0.000767	4438
	Sd	1.506724	29.97295	0.165083	0.372636	23.08391	0.000253	31.29945

Table 3: Schaffer Test Function for 5000 generations

Test Function : Schaffer								
Maximum Evaluations : 7000								
MOEA	**Metric**	**Spacing**	**S Metric**	**Generational Distance**	**Coverage**	**Coverage Difference**	**D Metric**	**Execution time**
NSGAII	Avg	0.06541486	13.95702	**0.004453362**	0.00968744	0.3820706	0.002795207	2770.3
	Min	0.0454264	13.9184	0.00355883	0	-0.464883	0.00232292	2641
	Max	0.0793565	13.9986	0.00494904	0.031746	1.46749	0.00353321	3109
	Sd	0.011669769	0.024454438	0.000483171	0.013088441	0.578286559	0.000340406	126.5921272
SPEA 2	Avg	0.06214031	14.35161	0.00486451	0.01961303	-0.1595968	0.002823951	**2737.6**
	Min	0.0456268	14.3136	0.00392023	0	-0.652608	0.00221091	2718
	Max	0.0944126	14.3919	0.00588534	0.0350877	0.463263	0.00321084	2797
	Sd	0.015567951	0.027644748	0.000736864	0.012419517	0.397240692	0.000305676	21.92005677
NSGAII*	Avg	**0.05750619**	**14.40338**	0.004833416	0.01087483	-0.04397301	0.002795495	18034.3
	Min	0.0462143	14.3653	0.00410278	0	-0.354676	0.00232195	12531
	Max	0.0743043	14.4456	0.00614926	0.0246914	0.539424	0.00347107	26641
	Sd	0.00888164	0.026420985	0.000624789	0.008248805	0.256647846	0.000336117	3817.48169
SPEA2*	Avg	2.168906	82.84095	0.106323	0.215202	9.719258	0.000281	6201.6
	Min	0	72.6429	0.070027	0	-21.6284	2.8E-05	6172
	Max	4.47206	89.3874	0.149983	0.5	25.4111	0.000607	6250
	Sd	1.75044	5.003878	0.024221	0.190513	13.52139	0.000205	33.2873

Table 4: Schaffer Test Function for 7000 generations

Test Function : Schaffer								
Maximum Evaluations : 12000								
MOEA	**Metric**	**Spacing**	**S Metric**	**Generational Distance**	**Coverage**	**Coverage Difference**	**D Metric**	**Execution time**
NSGAII	Avg	0.03467587	**13.87635**	0.0025483	0.006	0.005290931	0.00432087	**4509.5**
	Min	0.0260134	13.8618	0.00218351	0	-0.247208	0.00401089	4359
	Max	0.0486865	13.8838	0.00295501	0.01	0.396809	0.00466075	5125
	Sd	0.007393839	0.007417734	0.000243642	0.005163978	0.174123941	0.000177791	220.4471466
SPEA 2	Avg	**0.0317691**	13.72102	**0.002544732**	0.01	-0.060007254	0.004314792	5012.5
	Min	0.0251085	13.7118	0.00208738	0	-0.330261	0.00387698	4890
	Max	0.0381341	13.7323	0.00322549	0.02	0.172418	0.00470302	5156
	Sd	0.004528296	0.006874397	0.000323705	0.006666667	0.144643018	0.000235246	104.5086387
NSGAII*	Avg	0.06210616	13.75549	0.003739675	0.0081	0.01717511	0.004359609	31148.5
	Min	0.0479887	13.7444	0.00271907	0	-0.464457	0.00390946	19344
	Max	0.0771029	13.7665	0.0045945	0.02	0.408991	0.0047852	34859
	Sd	0.011067172	0.009088265	0.000555052	0.006190495	0.248039802	0.00025672	5362.730342
SPEA2*	Avg	1.869645	82.32969	0.086695	0.156984	0.344471	0.000352	11215.7
	Min	0.10341	72.6928	0.05173	0	-30.4027	6.63E-05	10907
	Max	5.75853	87.0451	0.137502	0.4	12.9799	0.000885	11750
	Sd	1.863832	4.722306	0.027877	0.169825	14.71341	0.00029	247.9113

Table 5: Schaffer Test Function for 12000 generations

6.5.4 ZDT1 Test Function - Summary of results

Test Function : ZDT1								
Maximum Evaluations : 5000								
MOEA	**Metric**	**Spacing**	**S Metric**	**Generational Distance**	**Coverage**	**Coverage Difference**	**D Metric**	**Execution time**
NSGAII	Avg	**0.01761894**	0.8997265	**0.003193552**	1	0.2146941	0	**1265.4**
	Min	0.0144158	0.802358	0.00243434	1	0.14513	0	1187
	Max	0.0224261	0.948946	0.00478327	1	0.35619	0	1312
	Sd	0.002735922	0.045864412	0.000728572	0	0.066130879	0	36.23135536
SPEA 2	Avg	0.02655061	1.0111216	0.004351858	1	0.309496	0	2357.6
	Min	0.0183806	0.97961	0.00284769	1	0.217936	0	2328
	Max	0.0437705	1.10006	0.00480402	1	0.409546	0	2407
	Sd	0.007833242	0.037047888	0.000608044	0	0.062072801	0	27.2282533
NSGAII*	Avg	0.02249254	0.9112688	0.004362227	1	0.2817175	0	1192.1
	Min	0.0175668	0.84305	0.00361584	1	0.21337	0	1141
	Max	0.0302852	0.96492	0.00546781	1	0.334171	0	1406
	Sd	0.004511111	0.035538508	0.00053585	0	0.036719123	0	76.84826319
SPEA2*	Avg	0.02133393	**1.1457124**	0.009098993	1	0.779746	0	5407.9
	Min	0.0150963	0.956024	0.00663923	1	0.575376	0	5125
	Max	0.053606	1.23984	0.01432	1	1.22484	0	5797
	Sd	0.011550531	0.077451023	0.002355142	0	0.17827152	0	288.4908818

Table 6: ZDT1 Test Function for 5000 generations

Test Function : ZDT1								
Maximum Evaluations : 7000								
MOEA	**Metric**	**Spacing**	**S Metric**	**Generational Distance**	**Coverage**	**Coverage Difference**	**D Metric**	**Execution time**
NSGAII	Avg	**0.010681607**	0.7809812	**0.001350462**	1	0.09273318	0	**1656.3**
	Min	0.00732323	0.744539	0.00101159	1	0.064281	0	1485
	Max	0.0139857	0.802795	0.00187628	1	0.148626	0	1844
	Sd	0.001783894	0.017064345	0.000251297	0	0.02392317	0	131.4094957
SPEA 2	Avg	0.01576466	**0.940221**	0.002243562	1	0.2039435	0	3329.7
	Min	0.0118067	0.894147	0.0016682	1	0.169401	0	3281
	Max	0.0231072	0.982718	0.00288223	1	0.282956	0	3437
	Sd	0.003765815	0.027955868	0.000383528	0	0.031035794	0	43.71892547
NSGAII*	Avg	0.012321582	0.7787452	0.001691574	1	0.11067947	0	1742.2
	Min	0.00813085	0.719903	0.00119937	1	0.0701497	0	1672
	Max	0.0152287	0.813097	0.00259483	1	0.159818	0	1938
	Sd	0.002594436	0.027826053	0.000418936	0	0.031472736	0	77.85713412
SPEA2*	Avg	0.012160202	0.6756576	0.008100803	1	0.6391736	0	7231.1
	Min	0.00938047	0.576582	0.00606761	1	0.512559	0	7187
	Max	0.0166163	0.734321	0.0101685	1	0.754125	0	7266
	Sd	0.002157113	0.047819997	0.001317958	0	0.075290859	0	35.2087426

Table 7: ZDT1 Test Function for 7000 generations

Test Function : ZDT1								
Maximum Evaluations : 12000								
MOEA	**Metric**	**Spacing**	**S Metric**	**Generational Distance**	**Coverage**	**Coverage Difference**	**D Metric**	**Execution time**
NSGAII	Avg	0.006664239	0.6756519	**0.000277038**	0.995	0.013501777	7.11461E-08	**2472**
	Min	0.00581581	0.671509	0.000242112	0.98	0.00842381	0	2468
	Max	0.00807983	0.678153	0.000340268	1	0.0200203	3.33241E-07	2485
	Sd	0.00064612	0.001891775	2.80328E-05	0.007071068	0.003508426	1.17671E-07	6.863753427
SPEA 2	Avg	0.005029205	**0.7169763**	0.000404536	1	0.0280154	0	6773.4
	Min	0.00403858	0.710018	0.000354129	1	0.0171999	0	6547
	Max	0.00638048	0.720535	0.000508522	1	0.0500253	0	7234
	Sd	0.000664308	0.003502124	5.20972E-05	0	0.010484237	0	189.6922654
NSGAII*	Avg	0.008170093	0.6864284	0.000331628	0.999	0.014800536	9.30431E-08	3573.3
	Min	0.00707532	0.680285	0.000279114	0.99	0.00741076	0	3437
	Max	0.00918807	0.690487	0.00042288	1	0.0222313	9.30431E-07	3875
	Sd	0.000772635	0.003471865	4.99123E-05	0.003162278	0.004854179	2.94228E-07	129.1838053
SPEA2*	Avg	**0.002567452**	0.3995725	0.01165386	1	0.617471	0	12970.3
	Min	0.00181475	0.331057	0.0104335	1	0.598091	0	12765
	Max	0.0031423	0.443068	0.0136027	1	0.639271	0	13312
	Sd	0.000439163	0.039412575	0.001158179	0	0.014036686	0	208.0213931

Table 8: ZDT1 Test Function for 12000 generations

6.5.5 ZDT3 Test Function - Summary of results

Test Function : ZDT3								
Maximum Evaluations : 5000								
MOEA	**Metric**	**Spacing**	**S Metric**	**Generational Distance**	**Coverage**	**Coverage Difference**	**D Metric**	**Execution time**
NSGAII	Avg	**0.0154456**	0.9908863	**0.002644349**	1	0.1860695	0	**1212.6**
	Min	0.0120711	0.924505	0.00224124	1	0.140058	0	1171
	Max	0.0186723	1.02582	0.0033245	1	0.251435	0	1250
	Sd	0.002036838	0.036071799	0.000392929	0	0.031241976	0	30.60210596
SPEA 2	Avg	0.02145701	**1.243168**	0.003557599	1	0.2767322	0	2361.1
	Min	0.0140933	1.18986	0.0030214	1	0.221539	0	2344
	Max	0.0424615	1.27944	0.00536689	1	0.593699	0	2407
	Sd	0.008294363	0.023721081	0.000683211	0	0.113172038	0	24.93302139
NSGAII*	Avg	0.017324	1.018238	0.003296	1	0.253051	0	1262.5
	Min	0.012122	0.950226	0.002206	1	0.170667	0	1203
	Max	0.024564	1.12317	0.004064	1	0.308099	0	1547
	Sd	0.004365	0.049075	0.000533	0	0.036881	0	100.7144
SPEA2*	Avg	0.020424	0.904882	0.005459	1	0.441293	0	5136
	Min	0.015377	0.805628	0.003277	1	0.233473	0	5109
	Max	0.033791	0.9995	0.00796	1	0.645393	0	5188
	Sd	0.005291	0.062017	0.001821	0	0.189984	0	24.67567

Table 9: ZDT3 Test Function for 5000 generations

Test Function : ZDT3								
Maximum Evaluations : 7000								
MOEA	**Metric**	**Spacing**	**S Metric**	**Generational Distance**	**Coverage**	**Coverage Difference**	**D Metric**	**Execution time**
NSGAII	Avg	**0.011111343**	0.9111428	**0.001096795**	1	0.08596119	0	**1579.7**
	Min	0.00749903	0.854473	0.000501127	1	0.0385018	0	1484
	Max	0.0219104	0.95776	0.00195407	1	0.143646	0	1828
	Sd	0.004197074	0.024975712	0.000355287	0	0.02641703	0	110.6275533
SPEA 2	Avg	0.013347	**0.972673**	0.001852	1	0.138973	0	3342.3
	Min	0.01086	0.899186	0.001032	1	0.08729	0	3297
	Max	0.018095	1.03562	0.002609	1	0.209713	0	3422
	Sd	0.002197	0.041743	0.000502	0	0.035491	0	41.16107
NSGAII*	Avg	0.011974	0.905334	0.001601	1	0.120919	0	1861
	Min	0.008475	0.865063	0.000813	1	0.065642	0	1797
	Max	0.016654	0.966835	0.002119	1	0.172974	0	2203
	Sd	0.002574	0.03252	0.000415	0	0.035321	0	120.8461
SPEA2*	Avg	0.01188	0.803219	0.003888	1	0.391777	0	7307.7
	Min	0.009037	0.730636	0.001699	1	0.123653	0	7187
	Max	0.015482	0.845455	0.005411	1	0.502598	0	7969
	Sd	0.002528	0.034273	0.001138	0	0.138462	0	234.688

Table 10: ZDT3 Test Function for 7000 generations

Test Function : ZDT3								
Maximum Evaluations : 12000								
MOEA	**Metric**	**Spacing**	**S Metric**	**Generational Distance**	**Coverage**	**Coverage Difference**	**D Metric**	**Execution time**
NSGAII	Avg	0.007379528	0.8053063	**0.00020948**	1	0.010720972	0	**2498.3**
	Min	0.00669593	0.80193	0.000191218	1	0.00372046	0	2484
	Max	0.0087414	0.808536	0.00022383	1	0.0158356	0	2547
	Sd	0.000612572	0.002726726	1.10671E-05	0	0.003649753	0	22.72076877
SPEA 2	Avg	0.005626	**0.826239**	0.000296	1	0.024431	0	7051.5
	Min	0.004636	0.820784	0.000257	1	0.019091	0	6812
	Max	0.006899	0.832328	0.000342	1	0.030909	0	7500
	Sd	0.000727	0.003973	2.97E-05	0	0.003986	0	214.4306
NSGAII*	Avg	0.007121	0.798703	0.000273	1	0.014717	0	3695.2
	Min	0.005345	0.791997	0.000216	1	0.00831	0	3562
	Max	0.008449	0.803722	0.000333	1	0.020838	0	3968
	Sd	0.001079	0.004004	4.13E-05	0	0.0044	0	140.1918
SPEA2*	Avg	**0.004221**	0.71477	0.002896	1	0.408872	0	13967.4
	Min	0.003002	0.655506	0.00071	1	0.170762	0	13407
	Max	0.006489	0.741127	0.006975	1	0.587711	0	14703
	Sd	0.000984	0.025849	0.001759	0	0.150909	0	420.1053

Table 11: ZDT3 Test Function for 12000 generations

6.6 Generated Pareto Fronts

The non dominated fronts generated for the different test functions are visualized below. All the obtained fronts are plotted against the true Pareto front. In this section only fronts which were generated at 10000 runs are shown. For more listing refer to Appendix A.

NSGAII **NSGAII*** **SPEA2** **SPEA2***

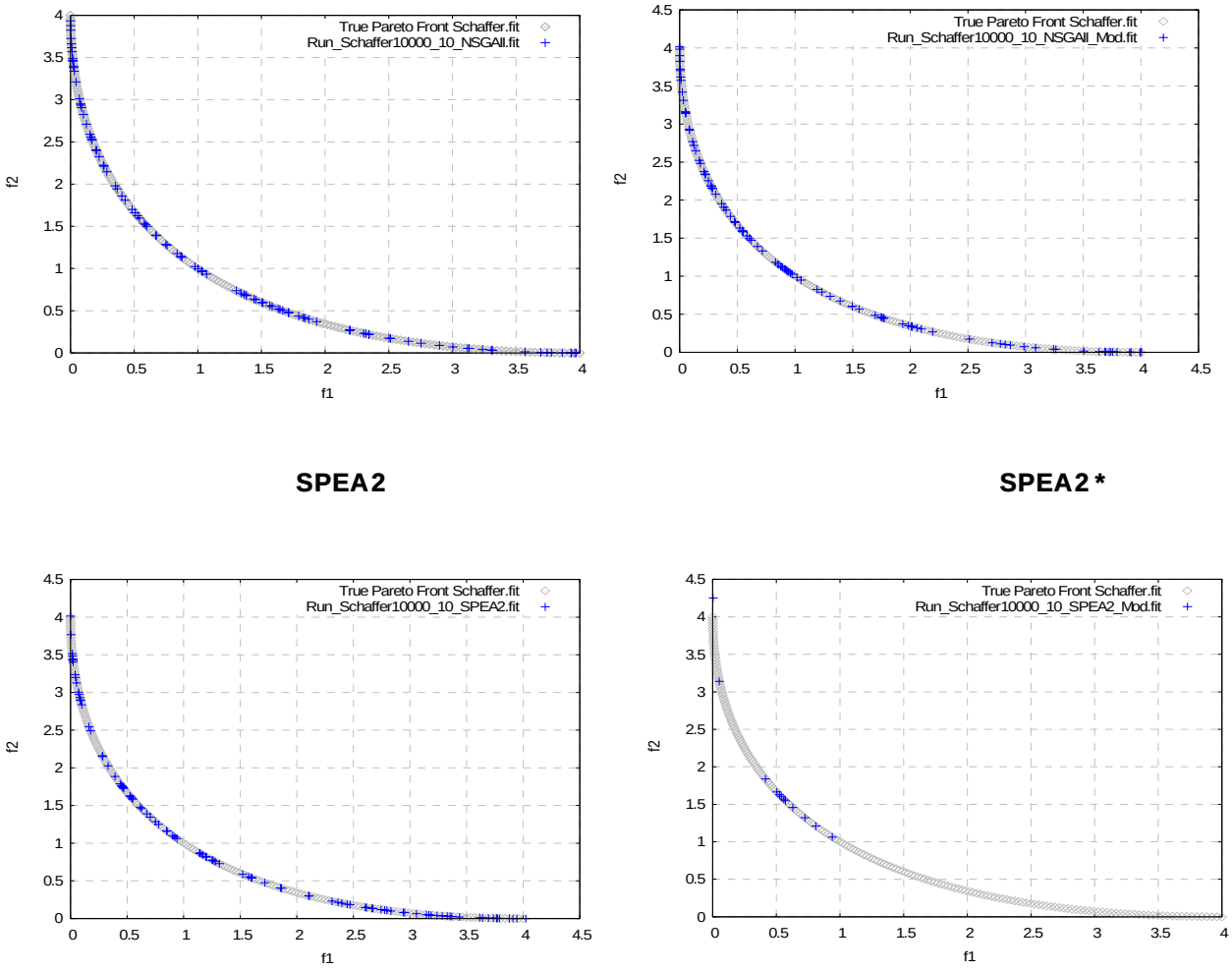


Figure 31: Schaffer Test Function Results

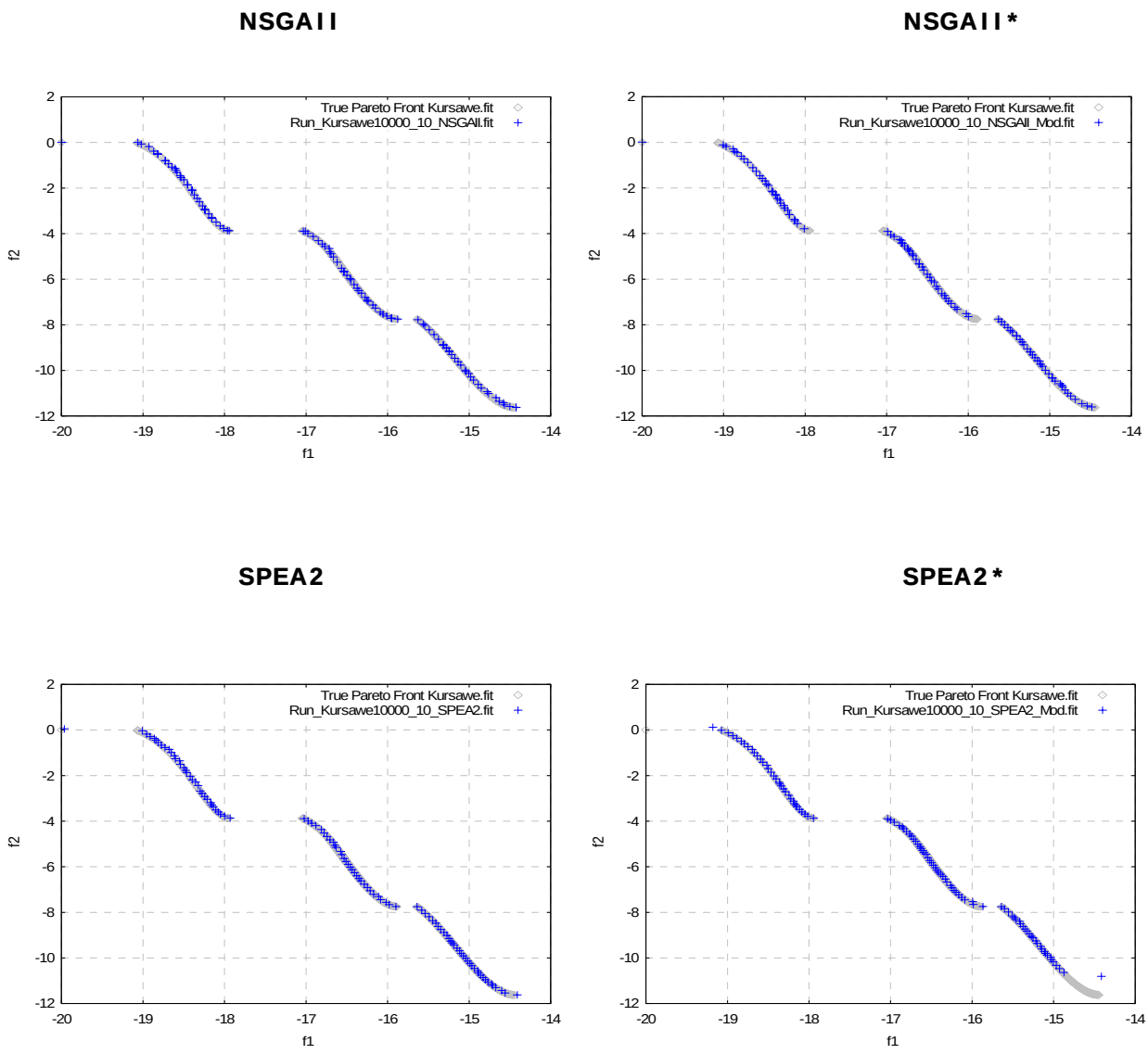


Figure 32: Kursawe Test Function Results

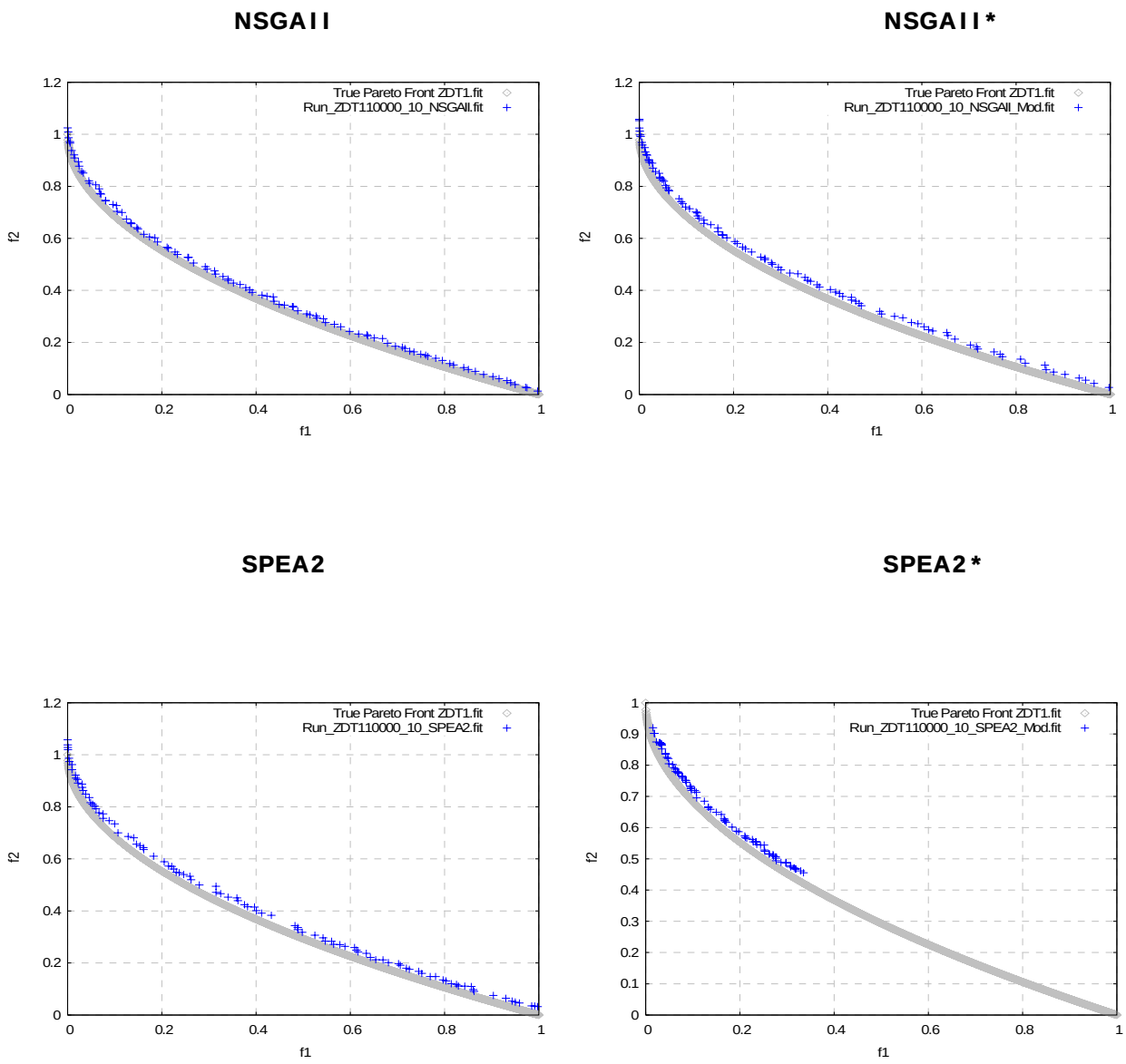


Figure 33: ZDT1 Test Function Results

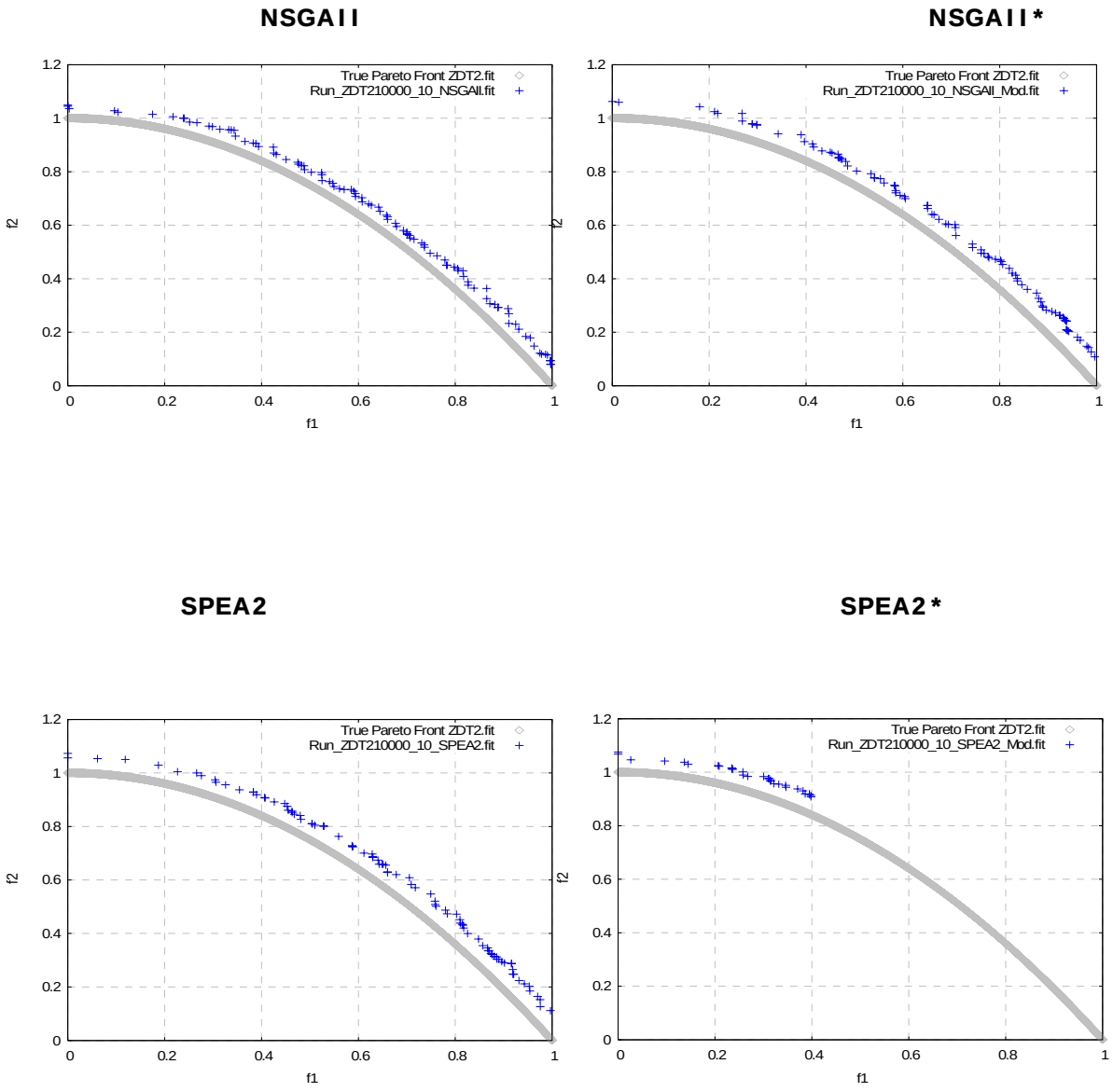


Figure 34: ZDT2 Test Function Results

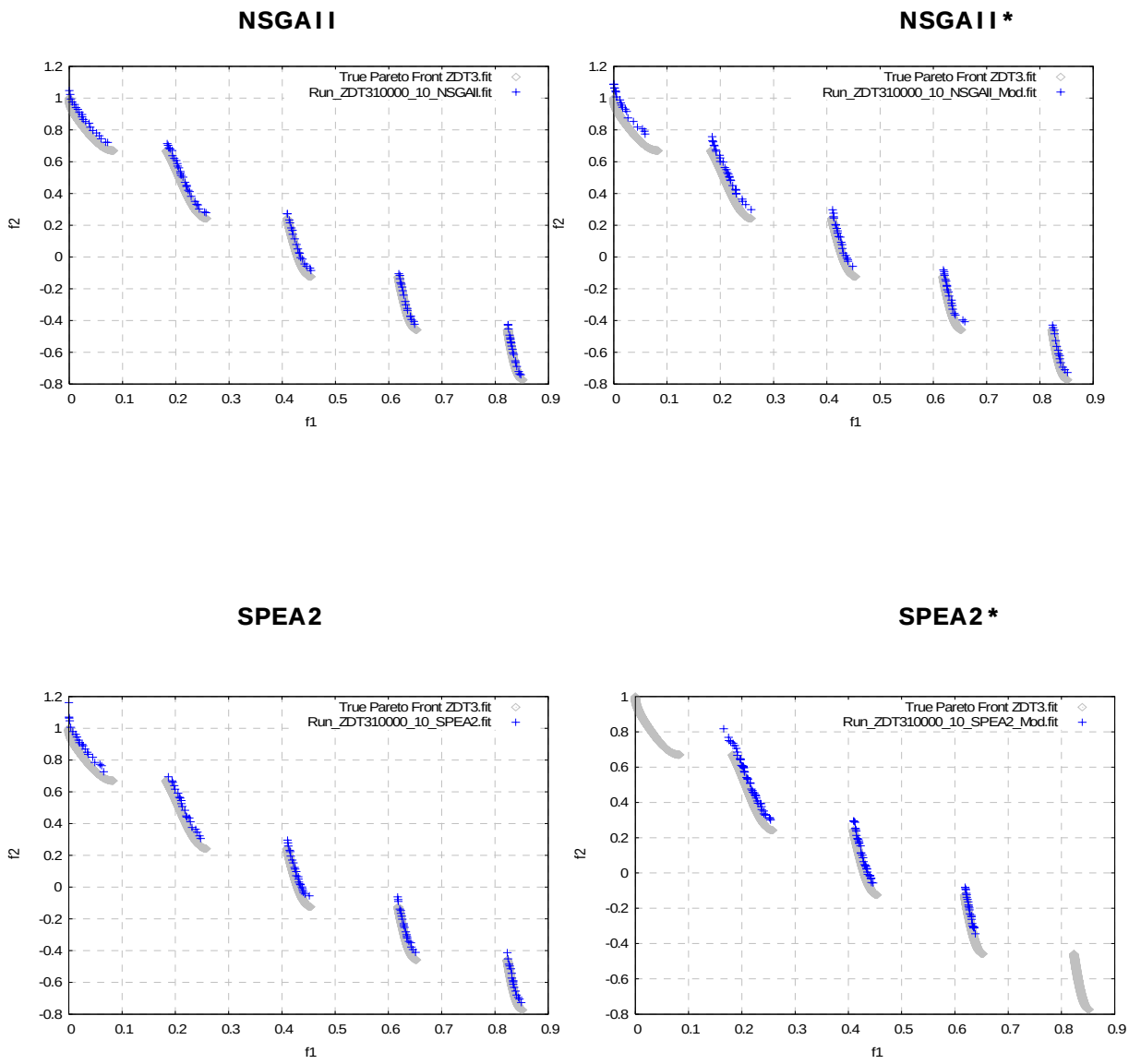


Figure 35: ZDT3 Test Function Results

Chapter 7

Summary and Conclusion

7.1 Summary

In this work several aspects of solving multi-objective problems by the use of evolutionary algorithms are analyzed. In particular, an inevitably prominent area of MOEA namely diversity maintenance, has been given prime importance in the discussion. Review on the state of the art in evolutionary algorithms for multi-objective optimization problems was presented. The importance of test functions and performance metrics are highlighted in Chapter 3. A new MOEA toolbox has been designed and developed. The toolbox is implemented with several features aiding research in MOEAs with implementation of a number of algorithms, design of the test problems, graphical analysis of results etc. Additionally, two new hybrid algorithms based on diversity maintenance has been proposed and discussed in Chapter 5.The statistical analysis and discussion on the performance comparison is provided in chapter 6.

7.2 Conclusion

The fundamental base of multi-objective evolutionary algorithms is provided by diversity preserving techniques. Most of the MOEA's, while following the EA's skeletal structure, has been essentially found to vary in their diversity maintaining methods. Among the several diversity methods, the most effective ones are chosen for comparison in this work. These methods applied to multi-objective problems has been analyzed and reviewed in this work. The impact of altering the diversity mechanisms in different algorithms and their consequent ability of finding better solutions are researched. Two new hybrid methods have been presented based on this study. The crowding distance diversity method used in the NSGAII has been used in the SPEA2 algorithm, replacing its existing diversity measure. Similarly the archive truncation method based on k^{th} nearest neighbor method used in SPEA2 has been applied to NSGAII's diversity method and tested. As a result of the tests, the hybrid methods, NSGAII* and SPEA2* showed improved results in certain test scenarios whereas produced satisfactory results otherwise. The algorithms

outperformed their original counterparts several times particularly in the case of increased number of maximum evaluations. With the implementation, testing and comparison of these hybrid methods acting as the backbone of this work, extensive statistical analysis is performed to provide support to the findings. The algorithms have been executed for several runs and changing parameter settings. They were tested on the test problems discussed and the comparison analysis was achieved with the use of several performance metrics. Additionally, the MOPT toolbox designed with a user friendly interface serves to provide an effective tool for researchers and students in the field of evolutionary algorithms to implement novel algorithms as well as to enable testing, visualizing and comparing several algorithms.

7.3 Future work

As part of future enhancements the algorithms could be tested for performance in the case of large number of objectives. Applying performance metrics which does not require knowledge of the true Pareto, into the runtime operation of an EA could be a potential future area of exploration. The spacing and hyper volume metrics could provide an ideal boosting factor in driving the algorithm towards effective solutions using this method.

References

[1] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1-16, 1995.

[2] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications", *Swiss Federal Institute of Technology (ETH)*1999.

[3] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257-271, 1999.

[4] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221-248, 1995.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[6] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," In *Proceedings of Evolutionary Methods for Design, Optimization, and Control*, pp. 95-100, 2002.

[7] H. Lu and G. G. Yen, "Rank-density-based multiobjective genetic algorithm and benchmark test function study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 325-343, 2003.

[8] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," In *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC'94)*, pp. 82-87, 1994.

[9] C.A.C. Coello, "An Updated Survey of GA-Based Multiobjective Optimization Techniques", *ACM Computing Surveys,* Vol. 32, No. 2, June 2000.

[10] J. D. Knowles, L. Thiele and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers", *TIK Report No. 214, Computer Engineering and Networks Laboratory,* February 2006.

[11] K. Deb, "Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems", *Evolutionary Computation Volume 7, Number 3*, 1999.

[12] J. Kennedy and R. Eberhart, "Particle Swarm Optimization". In *Proceedings of the Fourth IEEE International Conference on Neural Networks*, IEEE Service center, pp. 1942-1948, 1995.

[13] K.E. Parsopoulos and M.N. Vrahatis, "Particle Swarm Optimization Method in Multiobjective Problems". In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002)*, pp. 603-607, 2002.

[14] Jonathan E. Fieldsend, "Multi-objective particle swarm optimisation methods". *Technical Report #419, Department of Computer Science, University of Exeter*, March 2004.

[15] C. A. C. Coello and M.S. Lechunga. "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization". In *Proceedings of the 2002 Congess on Evolutionary Computation, part of the 2002 IEEE World Congress on Computational Intelligence*, pages 1051-1056, 2002. IEEE Press.

[16] C. A. C. Coello, "A Short Tutorial on Evolutionary Multiobjective Optimization", *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 21-40. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[17] Azarm, S., B. Reynolds, and S. Narayanan, "Comparison of Two Multiobjective Optimization Techniques with and within Genetic Algorithms," *CD-ROM Proceedings of the 25th ASME Design Automation Conference,* Paper No. DETC99/DAC-8584, 1999.

[18] N. Hallam , "MultiObjective Evolutionary Algorithms: State of the Art", Technical Report, TR 1122004, *The University of Nottingham, Malaysia Campus*, Dec 2004.

[19] D. E. Goldberg, & J. Richardson, "Genetic algorithms with sharing for multi-Modal function optimization," *in textit Proc. 2nd Int. Conf. on Genetic Algorithms*, pp.41-49, 1987.

[20] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," *Addison Wesley*, *Reading, Massachusetts*, 1989.

[21] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[22] E. Zitzler, M. Laumanns, S. Bleuler, "*A Tutorial on Evolutionary Multiobjective Optimization.*" Workshop on Multiple Objective Metaheuristics (MOMH 2002), Springer-Verlag, 2003.

[23] Fonseca, C. M. and Fleming, P. J. (1998)," Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A Unified Formulation." *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, 28(1):26–37.

[24] D. Van Veldhuizen and G. Lamont, "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art.", *Evolutionary Computation*, 8(2):125-147, 2000.

[25] Horn, J., Nafpliotis, N. and Goldberg, D. E. (1994). A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In Michalewicz, Z., editor, *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp. 82–87, IEEE Press.

[26] Zitzler, E.: 2002, "A tutorial on evolutionary multiobjective optimization", Invited Talk,*Workshop on Multi-Objective Meta-Heuristics*. 4–5 November 2002.

[27] 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.

[28] 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.

[29] Horn, J. and Nafpliotis, N.: "Multiobjective optimization using the niched Pareto genetic algorithm", *IlliGAL Report 93005, University of Illinois at Urbana-Champaign*, 1993.

[30] Corne, D. W., Jerram, N. R., Knowles, J. D. and Oates, M. J.: "PESA-II: Region-based selection in evolutionary multiobjective optimization", 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, pp. 283–290, 2001.

[31] Corne, D. W., Knowles, J. D. and Oates, M. J.: 2000, "The Pareto envelope-based selection algorithm for multiobjective optimization", in *M. Schoenauer, K. Deb, G.Rudolph, X. Yao, E. Lutton, J. J. Merelo and H.-P. Schwefel (eds), Proceedings of the Parallel Problem Solving from Nature VI Conference, Springer-Verlag*, pp. 839–848.

[32] M. Laumanns, L. Thiele, E. Zitzler, and K. Deb. "Archiving with guaranteed convergence and diversity in multi-objective optimization", *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 439-447,pp. 9-13 July 2002.

[33] Joshua D. Knowles and David W. Corne. "The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Pareto Multiobjective Optimisation", In *1999 Congress on Evolutionary Computation*, pages 98-105, July 1999.

[34] Zitzler E, Deb K, Thiele L, "Comparison of multiobjective evolutionary algorithms: empirical results", *Evolutionary Computation* 8(2):173–195, 2000.

[35] Zitzler E, Deb K, Thiele L, "Combining convergence and diversity in evolutionary multi-objective optimization", *Evolutionary Computation* 10(3):263–282, 2002.

[36] Joshua D. Knowles, "Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization", *PhD Thesis, The University of Reading, Reading, UK, January 2002.*

[37] David A. Van Veldhuizen, "Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.", *PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, May 1999.*

[38] Jason R. Schott, "Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization." *Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, May 1995.*

[39] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns and Eckart Zitzler, "Scalable Test Problems for Evolutionary Multi-Objective Optimization", *TIK-Report No.112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), July, 2001.*

[40] J.D. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms". *In J.J. Grefensttete, editor, Proc. of the First International Conference on Genetic Algorithms (ICGA), pages 93–100, 1987.*

[41] F. Kursawe, "A Variant of Evolution Strategies for Vector Optimization.", *In H.P. Schwefel and R. M¨nner, editors, Parallel Problem Solving for Nature, volume 496 of Lecture Notes in Computer Science, pages 193–197, Berlin, Germany, 1990.*

[42] Silverman, B. W. "*Density estimation for statistics and data analysis*.", London: Chapman and Hall, 1986.

[43] GUIMOO (Graphical User Interface Multi-Objective Optimization), available at: <*http://www.lifl.fr/OPAC/guimoo/*>

[44] K. Hirschen and M. Schäfer "A Study on Evolutionary Multi-Objective Optimization for Flow Geometry Design Computational Mechanics", *ISSN 0178-7675, Vol. 37/2, pp. 131-141, 2006*

[45] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb and Eckart Zitzler, "On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms", *Technical Report 108, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH- 8092, May 2001.*

[46] Joshua D. Knowles and David W. Corne, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy", *Evolutionary Computation*, Vol no. 8(2), pp. 149-172, 2000.

[47] Reyes, M., Coello Coello, C.A. "*Improving pso-based multi-objective optimization using crowding, mutation and epsilon-dominance".*Third International Conference on Evolutionary MultiCriterion Optimization, EMO 2005.

[48] P. Hajela and C. Y. Lin. "*Genetic search strategies in multicriterion optimal design.* Structural Optimization", Vol no.4, pp. 99-107, 1992.

[49] J. David Schaffer. "*Multiple Objective Optimization with Vector Evaluated Genetic Algorithms". In* Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pp. 93-100, 1985.*

[50] M. P. Fourman. "*Compaction of Symbolic Layout using Genetic Algorithms", In* Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pp. 141-153, 1985.*

APPENDIX

A. Obtained Pareto fronts

Pareto front for Schaffer Test Function

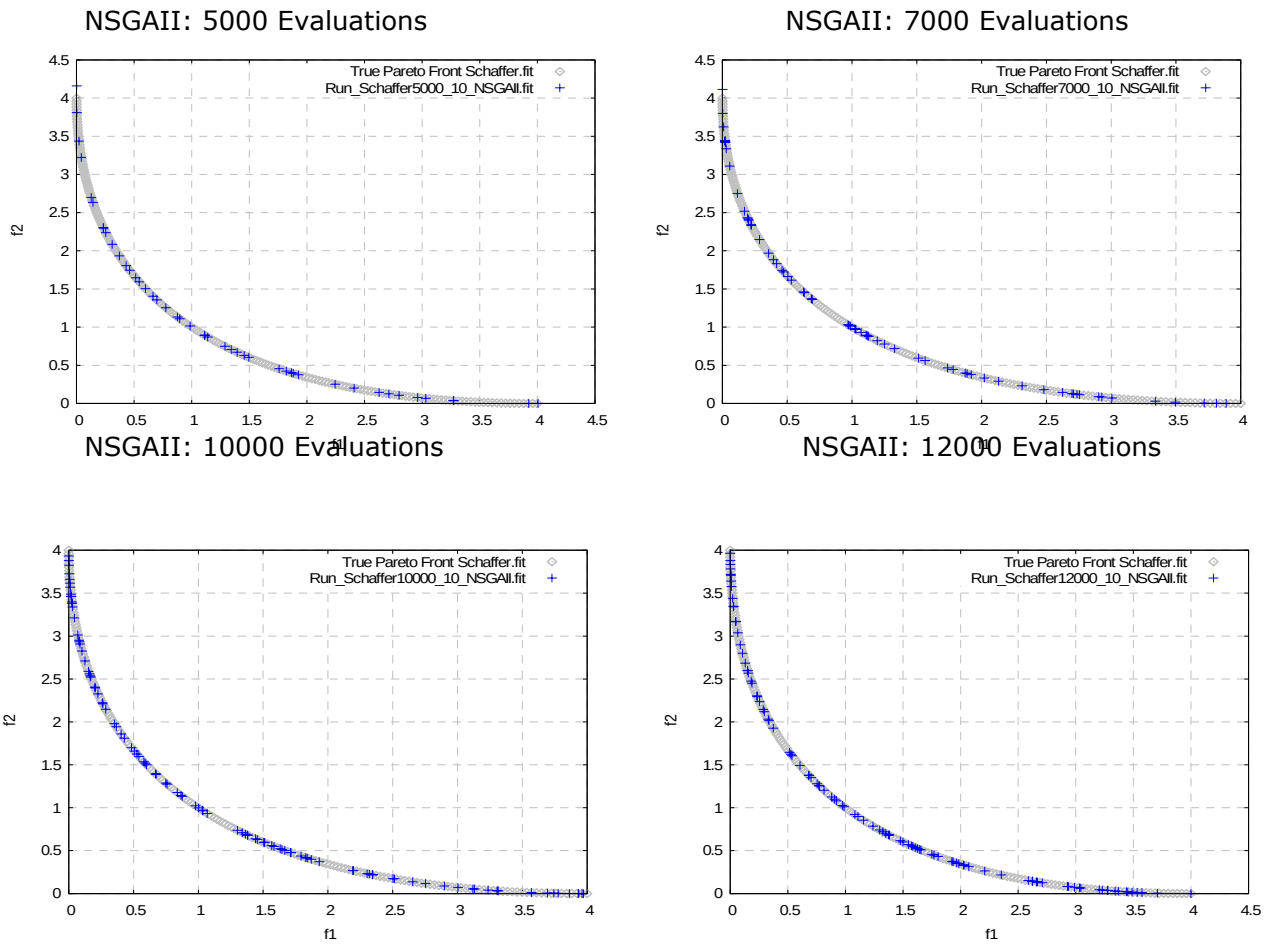


Figure 36: Pareto Fronts for Schaffer test function

B. Detailed results on Algorithm runs

Table 12: Detailed Results (SCH - 5000) NSGAII

Test Function	**: Schaffer**
Algorithm	**: NSGAII**
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.104905	14.7739	0.00766496	0.0227273	-0.0546837
2	0.123404	14.7977	0.00675159	0.0208333	0.197304
3	0.0495811	14.903	0.00665096	0.015625	0.218391
4	0.103515	14.7792	0.00702987	0.05	-0.456736
5	0.105848	14.8343	0.00685464	0.0222222	-0.770269
6	0.14227	14.8413	0.00770154	0.0243902	-0.792723
7	0.114779	14.7231	0.00748713	0.0263158	-0.262039
8	0.111856	14.782	0.00727949	0	0.204523
9	0.129357	14.8074	0.00608032	0.0243902	-0.403433
10	0.0830164	14.8712	0.00641005	0.0434783	-0.167129

Runs	D Metric	Execution Time
1	0.00171787	2250
2	0.00197715	2172
3	0.00306365	2156
4	0.00189935	2078
5	0.00207992	2125
6	0.00199078	2141
7	0.00172418	2204
8	0.0018323	2109
9	0.00150095	2109
10	0.00233123	2172

Table 13:Detailed Results (SCH - 7000) NSGAII

Test Function	**: Schaffer**
Algorithm	**: NSGAII**
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0793565	13.9474	0.00492845	0	1.46749
2	0.0639287	13.9845	0.00381773	0.025974	-0.256545
3	0.0501151	13.9696	0.0041551	0	0.861298
4	0.0763856	13.9266	0.00493053	0.031746	-0.464883
5	0.0671323	13.9184	0.0047593	0	0.6937
6	0.0748454	13.9684	0.0044807	0	0.608634
7	0.0640341	13.9546	0.00494904	0	0.546288
8	0.0454264	13.9986	0.00355883	0.0235294	-0.10403
9	0.0569989	13.9545	0.00439524	0	0.130811
10	0.0759256	13.9476	0.0045587	0.015625	0.337943

Runs	D Metric	Execution Time
1	0.00247326	2782
2	0.00287862	2719
3	0.00295233	2734
4	0.00247822	2781
5	0.00232292	2734
6	0.00292082	2781
7	0.00289287	2703
8	0.00353321	2641
9	0.00282057	2719
10	0.00267925	3109

Table 14:Detailed Results (SCH - 12000) NSGAII

Test Function	**: Schaffer**
Algorithm	**: NSGAII**
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0313061	13.8823	0.00247444	0	0.0440512
2	0.0321405	13.8777	0.00257278	0	0.114136
3	0.028535	13.8837	0.00227559	0	-0.00947189
4	0.0284426	13.8838	0.00218351	0	0.396809
5	0.0260134	13.8677	0.00264776	0.01	-0.0430155
6	0.0325723	13.8761	0.00236988	0.01	-0.167466
7	0.0357078	13.8815	0.00255969	0.01	-0.0624294
8	0.0450542	13.8707	0.002885	0.01	-0.0379925
9	0.0383003	13.8782	0.00255934	0.01	-0.247208
10	0.0486865	13.8618	0.00295501	0.01	0.0654964

Runs	D Metric	Execution Time
1	0.00428331	4407
2	0.00422487	5125
3	0.00431454	4484
4	0.00466075	4422
5	0.00431922	4407
6	0.00444359	4453
7	0.00441698	4359
8	0.00401089	4469
9	0.0043941	4469
10	0.00414045	4500

Table 15:Detailed Results (ZDT1 - 5000) NSGAII

Test Function	**: ZDT1**
Algorithm	**: NSGAII**
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0158019	0.948946	0.00243434	1	0.14513
2	0.0200079	0.892339	0.00335172	1	0.35619
3	0.0189391	0.926621	0.00265751	1	0.171547
4	0.0224261	0.802358	0.00478327	1	0.298509
5	0.0181124	0.920285	0.00289083	1	0.208317
6	0.014459	0.939849	0.0025885	1	0.161129
7	0.019837	0.858607	0.00375984	1	0.221731
8	0.0147809	0.873588	0.00359581	1	0.213094
9	0.0144158	0.942076	0.00255647	1	0.1637
10	0.0174093	0.892596	0.00331723	1	0.207594

Runs	D Metric	Execution Time
1	0	1187
2	0	1282
3	0	1265
4	0	1234
5	0	1312
6	0	1265
7	0	1297
8	0	1250
9	0	1297
10	0	1265

Table 16:Detailed Results (ZDT1 - 7000) NSGAII

Test Function	**: ZDT1**
Algorithm	**: NSGAII**
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.00974308	0.768028	0.00137176	1	0.0968511
2	0.0105295	0.802743	0.00101159	1	0.064281
3	0.00732323	0.786174	0.00128497	1	0.0749446
4	0.00938536	0.787434	0.0012532	1	0.0765765
5	0.0115137	0.781462	0.0013587	1	0.0962217
6	0.010372	0.785042	0.0013063	1	0.0975538
7	0.0103267	0.802795	0.00102475	1	0.0713605
8	0.0122437	0.744539	0.00187628	1	0.148626
9	0.0139857	0.77153	0.0014598	1	0.0980766
10	0.0113931	0.780065	0.00155727	1	0.10284

Runs	D Metric	Execution Time
1	0	1516
2	0	1562
3	0	1735
4	0	1531
5	0	1844
6	0	1844
7	0	1640
8	0	1687
9	0	1719
10	0	1485

Table 17:Detailed Results (ZDT1 - 12000) NSGAII

Test Function	**: ZDT1**
Algorithm	**: NSGAII**
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.00609015	0.675238	0.000282582	0.99	0.0133547
2	0.00707092	0.671509	0.000340268	0.99	0.0164732
3	0.00668261	0.675887	0.000273265	1	0.0141712
4	0.00581581	0.67534	0.000280666	1	0.0200203
5	0.00807983	0.675747	0.000273605	1	0.0101185
6	0.00640227	0.675862	0.000273179	0.99	0.0136264
7	0.00631265	0.678153	0.000242112	1	0.00842381
8	0.00661258	0.674266	0.000297946	0.98	0.0148901
9	0.00718324	0.678074	0.000243055	1	0.00922006
10	0.00639233	0.676443	0.000263701	1	0.0147195

Runs	D Metric	Execution Time
1	1.24399e-007	2485
2	0	2469
3	2.17737e-007	2469
4	0	2468
5	0	2469
6	3.6084e-008	2469
7	0	2468
8	3.33241e-007	2469
9	0	2469
10	0	2485

Table 18:Detailed Results (ZDT3 - 5000) NSGAII

Test Function	**: ZDT3**
Algorithm	**: NSGAII**
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0167772	0.937186	0.00320283	1	0.212194
2	0.017511	0.924505	0.0033245	1	0.251435
3	0.015539	1.01855	0.0023201	1	0.172574
4	0.0139827	1.00666	0.00251036	1	0.162163
5	0.016519	1.01885	0.00230655	1	0.171263
6	0.0147573	0.994052	0.00265427	1	0.184867
7	0.0120711	1.00849	0.00243881	1	0.177096
8	0.013056	1.01039	0.00243886	1	0.178482
9	0.0155704	0.96436	0.00300597	1	0.210563
10	0.0186723	1.02582	0.00224124	1	0.140058

Runs	D Metric	Execution Time
1	0	1219
2	0	1172
3	0	1235
4	0	1234
5	0	1219
6	0	1250
7	0	1250
8	0	1188
9	0	1171
10	0	1188

Table 19:Detailed Results (ZDT3 - 7000) NSGAII

Test Function	**: ZDT3**
Algorithm	**: NSGAII**
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0102501	0.910574	0.00106016	1	0.092851
2	0.0119154	0.917687	0.00101073	1	0.07583
3	0.0126303	0.904929	0.00116683	1	0.0925134
4	0.0219104	0.854473	0.00195407	1	0.143646
5	0.00904512	0.908005	0.00111688	1	0.102108
6	0.0115105	0.907227	0.00113587	1	0.0812148
7	0.00755944	0.913174	0.00106817	1	0.0769561
8	0.00833604	0.918048	0.00100195	1	0.0751218
9	0.0104571	0.919551	0.000952166	1	0.080869
10	0.00749903	0.95776	0.000501127	1	0.0385018

Runs	D Metric	Execution Time
1	0	1656
2	0	1828
3	0	1625
4	0	1500
5	0	1500
6	0	1657
7	0	1500
8	0	1516
9	0	1484
10	0	1531

Table 20: Detailed Results (ZDT3 - 12000) NSGAII

Test Function	**: ZDT3**
Algorithm	**: NSGAII**
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.00707588	0.806506	0.000208021	1	0.0157171
2	0.00704275	0.808175	0.000191218	1	0.00734723
3	0.00670964	0.808536	0.000200373	1	0.011039
4	0.00737362	0.802709	0.000215285	1	0.0115012
5	0.00786111	0.802175	0.00022383	1	0.0113726
6	0.0087414	0.807663	0.000196337	1	0.0108824
7	0.00669593	0.80193	0.000222746	1	0.0158356
8	0.00711917	0.803532	0.000209318	1	0.0116085
9	0.00750347	0.803749	0.000208472	1	0.00372046
10	0.00767231	0.808088	0.000219195	1	0.00818563

Runs	D Metric	Execution Time
1	0	2485
2	0	2531
3	0	2484
4	0	2484
5	0	2484
6	0	2500
7	0	2500
8	0	2547
9	0	2484
10	0	2484

Table 21:Detailed Results (SCH - 5000) NSGAII*

Test Function	**: Schaffer**
Algorithm	**: NSGAII***
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0745312	14.0892	0.0048623	0.0357143	-0.57158
2	0.128458	14.0498	0.00734218	0.0181818	-0.0345402
3	0.0787597	14.008	0.0076099	0.0212766	0.747914
4	0.0680044	14.056	0.00613411	0.0357143	-0.21149
5	0.0844249	14.002	0.00660159	0	-0.0157843
6	0.0828549	14.0468	0.00674431	0	1.4067
7	0.130139	14.0396	0.00654536	0	0.212811
8	0.122523	13.9572	0.00740185	0.0465116	-0.34479
9	0.101218	14.0761	0.00502374	0.0196078	0.0799713
10	0.0859192	14.1019	0.00477164	0.0377358	-0.745261

Runs	D Metric	Execution Time
1	0.00220179	9281
2	0.00247652	9297
3	0.00207723	10110
4	0.00224362	11719
5	0.00224362	11359
6	0.00173272	11421
7	0.00201354	11485
8	0.00139909	11531
9	0.00230332	12031
10	0.00222442	11703

Table 22:Detailed Results (SCH - 7000) NSGAII*

Test Function	**: Schaffer**
Algorithm	**: NSGAII***
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0462143	14.4068	0.00410278	0.0136986	0.175236
2	0.0609568	14.4456	0.00449622	0.0131579	-0.293289
3	0.0662255	14.4141	0.0046446	0.0142857	-0.197252
4	0.0462775	14.4094	0.00429026	0.0125	-0.0785666
5	0.0577404	14.3653	0.00520086	0.016129	-0.0885754
6	0.0576854	14.3737	0.00614926	0	-0.083519
7	0.0516139	14.4025	0.00504661	0.0142857	-0.121549
8	0.0620705	14.3734	0.0054223	0	0.0630369
9	0.0519733	14.4065	0.00463444	0	0.539424
10	0.0743043	14.4365	0.00434683	0.0246914	-0.354676

Runs	D Metric	Execution Time
1	0.00290559	12531
2	0.00301749	14094
3	0.0029318	15578
4	0.00296446	17328
5	0.00268319	26641
6	0.00263613	17797
7	0.00239241	19453
8	0.00232195	19093
9	0.00263086	18937
10	0.00347107	18891

Table 23: Detailed Results (SCH - 12000) NSGAII*

Test Function	**: Schaffer**
Algorithm	**: NSGAII***
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.061509	13.7597	0.00351299	0	0.283363
2	0.0482078	13.7466	0.00343338	0.01	-0.0896702
3	0.0770399	13.7665	0.0045945	0.001	-0.140594
4	0.0771029	13.7648	0.00448271	0.02	0.207424
5	0.0707345	13.7589	0.00378852	0.01	0.408991
6	0.0568463	13.764	0.00330388	0	0.0987568
7	0.050804	13.7444	0.00372053	0.01	-0.101853
8	0.0631941	13.7454	0.00377066	0.01	-0.0464067
9	0.0676344	13.7449	0.00407051	0.01	-0.464457
10	0.0479887	13.7597	0.00271907	0.01	0.0161972

Runs	D Metric	Execution Time
1	0.00452139	19344
2	0.00405282	23125
3	0.00436539	32125
4	0.0047852	34188
5	0.00448036	34859
6	0.00450744	33297
7	0.00450212	32984
8	0.00426538	33672
9	0.00390946	34703
10	0.00420653	33188

Table 24: Detailed Results (ZDT1 - 5000) NSGAII*

Test Function	**: ZDT1**
Algorithm	**: NSGAII***
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0213005	0.900051	0.0045078	1	0.288017
2	0.0232447	0.941774	0.00396193	1	0.297854
3	0.0274935	0.869505	0.00493183	1	0.290331
4	0.0241169	0.84305	0.00546781	1	0.31424
5	0.0193843	0.929929	0.00403534	1	0.255859
6	0.0176238	0.923815	0.00421553	1	0.30569
7	0.0302852	0.907269	0.00437182	1	0.281049
8	0.0175668	0.931419	0.00398479	1	0.236594
9	0.0261132	0.900956	0.00452958	1	0.334171
10	0.0177965	0.96492	0.00361584	1	0.21337

Runs	D Metric	Execution Time
1	0	1187
2	0	1141
3	0	1172
4	0	1187
5	0	1172
6	0	1406
7	0	1156
8	0	1156
9	0	1156
10	0	1188

Table 25:Detailed Results (ZDT1- 7000) NSGAII*

Test Function	**: ZDT1**
Algorithm	**: NSGAII***
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.01503	0.784044	0.00159946	1	0.0954626
2	0.00993597	0.80346	0.00133421	1	0.0888442
3	0.0137079	0.764767	0.00189791	1	0.128646
4	0.014192	0.719903	0.00259483	1	0.154019
5	0.0152287	0.808596	0.00124948	1	0.0701497
6	0.0129178	0.766742	0.0018799	1	0.159818
7	0.011847	0.759259	0.00197522	1	0.136495
8	0.0136244	0.787402	0.00151608	1	0.0931162
9	0.0086012	0.780182	0.00166928	1	0.098121
10	0.00813085	0.813097	0.00119937	1	0.082123

Runs	D Metric	Execution Time
1	0	1718
2	0	1750
3	0	1672
4	0	1938
5	0	1765
6	0	1703
7	0	1719
8	0	1672
9	0	1703
10	0	1782

Table 26: Detailed Results (ZDT1- 12000) NSGAII*

Test Function	**: ZDT1**
Algorithm	**: NSGAII***
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.00707532	0.688378	0.000305681	1	0.0203528
2	0.00730712	0.680285	0.00042288	1	0.0187378
3	0.00866543	0.68899	0.000289497	1	0.0103549
4	0.00896427	0.686255	0.000336969	1	0.0134582
5	0.00918807	0.686881	0.000321839	1	0.0123579
6	0.00815082	0.684295	0.000360295	1	0.018665
7	0.00808879	0.689308	0.000293941	1	0.0124542
8	0.00859897	0.681176	0.000405972	1	0.0222313
9	0.0070907	0.690487	0.000279114	1	0.00741076
10	0.00857144	0.688229	0.000300089	0.99	0.0119825

Runs	D Metric	Execution Time
1	0	3500
2	0	3640
3	0	3688
4	0	3547
5	0	3531
6	0	3484
7	0	3875
8	0	3437
9	9.30431e-007	3531
10	0	3500

Table 27: Detail Results (ZDT3 - 5000) NSGAII*

Test Function	**: ZDT3**
Algorithm	**: NSGAII***
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0190038	0.950226	0	1	0.308099
2	0.0121969	0.993583	0.0050702	1	0.255438
3	0.0121218	1.03031	0.0125625	1	0.255645
4	0.0170949	1.12317	0.0133915	1	0.170667
5	0.0215041	0.981719	0.0112446	1	0.289643
6	0.0222327	0.983505	0.0052517	1	0.253188
7	0.0139222	1.04932	0.0129204	1	0.242768
8	0.0161655	1.00541	0.00504821	1	0.265786
9	0.0245642	1.0571	0.00723334	1	0.226423
10	0.0144339	1.00804	0.00627098	1	0.26285

Runs	D Metric	Execution Time
1	0	1547
2	0	1234
3	0	1234
4	0	1234
5	0	1234
6	0	1219
7	0	1250
8	0	1235
9	0	1235
10	0	1203

Table 28: Detail Results (ZDT3- 7000) NSGAII*

Test Function	**:** ZDT3
Algorithm	**: NSGAII***
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0166542	0.883421	0.0018585	1	0.155383
2	0.0121641	0.883945	0.00189823	1	0.144119
3	0.0132729	0.892794	0.00175246	1	0.126819
4	0.00902107	0.913015	0.0014939	1	0.101395
5	0.0142743	0.878422	0.00193849	1	0.144092
6	0.00923166	0.966835	0.000813139	1	0.0656416
7	0.0117386	0.938879	0.00118266	1	0.0814288
8	0.0115328	0.894214	0.00175569	1	0.130163
9	0.0084745	0.936747	0.00120245	1	0.087173
10	0.0133798	0.865063	0.00211941	1	0.172974

Runs	D Metric	Execution Time
1	0	1829
2	0	1797
3	0	1844
4	0	1828
5	0	1813
6	0	2203
7	0	1828
8	0	1828
9	0	1828
10	0	1812

Table 29: Detail Results (ZDT3- 12000) NSGAII*

Test Function	**:** ZDT3
Algorithm	**: NSGAII***
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0084487	0.803476	0.000251508	1	0.00830966
2	0.00574161	0.798377	0.000267878	1	0.0146495
3	0.00795096	0.803722	0.000216081	1	0.00912201
4	0.00611749	0.802025	0.000245702	1	0.0125332
5	0.00770546	0.795912	0.000329541	1	0.0149168
6	0.00751448	0.794361	0.000332832	1	0.0188774
7	0.00812976	0.791997	0.000316306	1	0.0206831
8	0.00665044	0.801986	0.000226437	1	0.0121536
9	0.00534519	0.797187	0.000275848	1	0.0208377
10	0.00760576	0.79799	0.000265293	1	0.0150829

Runs	D Metric	Execution Time
1	0	3625
2	0	3688
3	0	3968
4	0	3672
5	0	3641
6	0	3609
7	0	3609
8	0	3641
9	0	3937
10	0	3562

Table 30: Detail Results (SCH - 5000) SPEA 2

Test Function	**: Schaffer**
Algorithm	**: SPEA 2**
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.116793	14.2686	0.00676068	0.0392157	-0.581295
2	0.102299	14.2866	0.00544966	0.0377358	-0.19869
3	0.101131	14.257	0.00667089	0.0212766	-0.651938
4	0.108457	14.1911	0.00911255	0.025	-0.420989
5	0.105932	14.2456	0.00746688	0.0208333	-0.0731812
6	0.112966	14.2937	0.00560185	0.0217391	-0.0504303
7	0.101588	14.2548	0.00633232	0.0196078	0.324505
8	0.0884626	14.2609	0.00715907	0.0192308	-0.0590315
9	0.0744838	14.1668	0.00815725	0.0227273	0.0252295
10	0.0928174	14.2307	0.0077131	0.05	-0.560721

Runs	D Metric	Execution Time
1	0.00229137	1984
2	0.00183601	1922
3	0.00194363	1968
4	0.00147822	1984
5	0.00201182	1938
6	0.00173411	1984
7	0.00193964	2000
8	0.00203899	1906
9	0.00180052	1984
10	0.00127214	1984

Table 31: Detail Results (SCH - 7000) SPEA 2

Test Function	**: Schaffer**
Algorithm	**: SPEA 2**
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0944126	14.3195	0.00588534	0.0181818	-0.652608
2	0.0572939	14.3416	0.00423929	0.0140845	0.198803
3	0.062646	14.3136	0.00558612	0.030303	-0.45076
4	0.0502236	14.321	0.00495588	0.028169	-0.328092
5	0.0503809	14.3919	0.00392023	0	0.463263
6	0.0517618	14.3726	0.00538223	0	0.266912
7	0.0456268	14.3683	0.00547603	0.027027	-0.625355
8	0.0831178	14.3415	0.00510385	0.0350877	-0.19222
9	0.0603361	14.3738	0.00406458	0.0285714	-0.408814
10	0.0656036	14.3723	0.00403155	0.0147059	0.132903

Runs	D Metric	Execution Time
1	0.00268358	2735
2	0.00281045	2734
3	0.00253911	2718
4	0.00292966	2734
5	0.00321084	2797
6	0.00317874	2734
7	0.002951	2735
8	0.00221091	2719
9	0.00303325	2735
10	0.00269197	2735

Table 32: Detail Results (SCH - 12000) SPEA 2

Test Function	**: Schaffer**
Algorithm	**: SPEA 2**
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0251085	13.7239	0.0025004	0.02	-0.12824
2	0.0369027	13.7118	0.0026001	0	0.172418
3	0.0278823	13.719	0.002624	0	-0.025425
4	0.0381341	13.7174	0.00255852	0.01	-0.15643
5	0.0349253	13.7323	0.00208738	0.01	0.0376768
6	0.0323894	13.7252	0.00232715	0.01	-0.330261
7	0.0264353	13.731	0.00223852	0.01	-0.00123024
8	0.0340224	13.714	0.00322549	0.02	-0.170024
9	0.0283978	13.7183	0.00242086	0.01	-0.0817719
10	0.0334932	13.7173	0.0028649	0.01	0.0832148

Runs	D Metric	Execution Time
1	0.00387698	5047
2	0.00446847	4890
3	0.00470302	5156
4	0.00415946	5032
5	0.00416654	4906
6	0.00441768	5000
7	0.00418099	5141
8	0.00438553	4891
9	0.00425373	5125
10	0.00453552	4937

Table 33: Detail Results (ZDT1 - 5000) SPEA 2

Test Function	**: ZDT1**
Algorithm	**: SPEA 2**
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0437705	1.00671	0.00447802	1	0.296132
2	0.028415	1.04558	0.00380171	1	0.282034
3	0.0210605	0.97961	0.00458859	1	0.275797
4	0.0183806	1.01256	0.00427008	1	0.264921
5	0.0226931	0.989598	0.00462164	1	0.303142
6	0.0201444	0.981096	0.00480402	1	0.28175
7	0.0311652	1.00158	0.00478962	1	0.409546
8	0.0239255	1.01132	0.00454334	1	0.403623
9	0.0337713	0.983102	0.00477387	1	0.360079
10	0.02218	1.10006	0.00284769	1	0.217936

Runs	D Metric	Execution Time
1	0	2343
2	0	2343
3	0	2328
4	0	2343
5	0	2407
6	0	2344
7	0	2359
8	0	2344
9	0	2359
10	0	2406

Table 34: Detail Results (ZDT1 - 7000) SPEA 2

Test Function	**: ZDT1**
Algorithm	**: SPEA 2**
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0205802	0.894147	0.00288223	1	0.204401
2	0.0122221	0.951118	0.00206945	1	0.197386
3	0.0132753	0.982718	0.0016682	1	0.193981
4	0.0140367	0.974075	0.00178486	1	0.216331
5	0.0231072	0.920961	0.00239455	1	0.180584
6	0.0138332	0.959957	0.00193944	1	0.186482
7	0.0157789	0.922267	0.00251142	1	0.210527
8	0.0118067	0.944897	0.00238319	1	0.282956
9	0.0144647	0.938938	0.00219139	1	0.169401
10	0.0185416	0.913132	0.00261089	1	0.197386

Runs	D Metric	Execution Time
1	0	3329
2	0	3297
3	0	3344
4	0	3297
5	0	3344
6	0	3281
7	0	3343
8	0	3313
9	0	3437
10	0	3312

Table 35: Detail Results (ZDT1 - 12000) SPEA 2

Test Function	**: ZDT1**
Algorithm	**: SPEA 2**
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.00403858	0.718849	0.000379719	1	0.0357524
2	0.00462824	0.720535	0.000354129	1	0.0207268
3	0.0048509	0.715096	0.000434961	1	0.0286587
4	0.00505082	0.719818	0.000363698	1	0.0282276
5	0.00638048	0.713568	0.000455372	1	0.0376999
6	0.00541078	0.710018	0.000508522	1	0.0500253
7	0.00510555	0.720106	0.000354286	1	0.0171999
8	0.00518839	0.71844	0.000381715	1	0.0182736
9	0.00541347	0.714382	0.00043984	1	0.0238366
10	0.00422484	0.718951	0.000373115	1	0.0197532

Runs	D Metric	Execution Time
1	0	6734
2	0	6782
3	0	6797
4	0	7234
5	0	6781
6	0	6609
7	0	6812
8	0	6547
9	0	6610
10	0	6828

Table 36: Detail Results (ZDT3 - 5000) SPEA 2

Test Function	**: ZDT3**
Algorithm	**: SPEA 2**
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0246232	1.18986	0.00392201	1	0.27558
2	0.0170117	1.24303	0.00342185	1	0.233451
3	0.0192198	1.25198	0.00323731	1	0.235824
4	0.019235	1.22105	0.00359773	1	0.279935
5	0.0207529	1.27944	0.0030214	1	0.228549
6	0.0140933	1.2396	0.00536689	1	0.593699
7	0.0424615	1.25501	0.00334453	1	0.229279
8	0.0257008	1.24674	0.00321468	1	0.227757
9	0.0155616	1.25303	0.0032287	1	0.221539
10	0.0159103	1.25194	0.00322089	1	0.241709

Runs	D Metric	Execution Time
1	0	2360
2	0	2406
3	0	2344
4	0	2344
5	0	2344
6	0	2359
7	0	2344
8	0	2407
9	0	2359
10	0	2344

Table 37: Detail Results (ZDT3 - 7000) SPEA 2

Test Function	**: ZDT3**
Algorithm	**: SPEA 2**
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.0140404	1.03362	0.00108192	1	0.0872904
2	0.0133935	0.984692	0.00174249	1	0.127757
3	0.0115782	1.03562	0.00103178	1	0.0942835
4	0.0108595	0.991001	0.00164605	1	0.12276
5	0.011635	0.899186	0.00260891	1	0.209713
6	0.0180954	0.958617	0.00201543	1	0.157353
7	0.0159193	0.952468	0.00214844	1	0.14529
8	0.0124187	0.939585	0.00228078	1	0.16662
9	0.0128448	0.978896	0.00183845	1	0.131176
10	0.0126806	0.95304	0.00212512	1	0.147487

Runs	D Metric	Execution Time
1	0	3344
2	0	3344
3	0	3422
4	0	3297
5	0	3390
6	0	3359
7	0	3297
8	0	3344
9	0	3297
10	0	3329

Table 38: Detail Results (ZDT3 - 12000) SPEA 2

Test Function	**: ZDT3**
Algorithm	**: SPEA 2**
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	0.00582075	0.825452	0.000291234	1	0.0235764
2	0.00689939	0.827478	0.000294602	1	0.0248355
3	0.00535329	0.82449	0.000303431	1	0.0285415
4	0.00580876	0.822844	0.000324582	1	0.0290183
5	0.00630802	0.821375	0.000342136	1	0.0309091
6	0.00560296	0.828474	0.000273501	1	0.0201061
7	0.00463634	0.827646	0.000277388	1	0.0236099
8	0.00500282	0.832328	0.000256674	1	0.0206808
9	0.00614415	0.83152	0.000264657	1	0.0190905
10	0.00468227	0.820784	0.000335294	1	0.0239441

Runs	D Metric	Execution Time
1	0	6907
2	0	7062
3	0	7500
4	0	7031
5	0	6828
6	0	7297
7	0	7094
8	0	6812
9	0	6906
10	0	7078

Table 39: Detail Results (SCH - 5000) SPEA 2*

Test Function	**: Schaffer**
Algorithm	**: SPEA 2***
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	2.98092	74.5917	0.0907294	0.333333	-23.5602
2	0	9.38501	0.452532	1	-31.939
3	1.0204	76.5101	0.0845583	0.0833333	12.0409
4	3.65446	71.2795	0.118141	0.5	-22.2403
5	0	0	0.495977	1	-38.1212
6	3.40208	73.8847	0.0958613	0.25	17.2947
7	0	68.7544	0.124789	0.5	24.2354
8	0.343925	79.6865	0.0512265	0	10.4234
9	0.332712	77.8671	0.0668402	0	11.6448
10	0.456301	78.5934	0.0655361	0.142857	9.54765

Runs	D Metric	Execution Time
1	0.000276087	4422
2	0	4359
3	0.000767394	4422
4	0.000141417	4422
5	0	4359
6	0.000257327	4422
7	9.85647e-005	4407
8	0.000637565	4359
9	0.000327574	4438
10	0.000297652	4421

Table 40: Detail Results (SCH - 7000) SPEA 2*

Test Function	**: Schaffer**
Algorithm	**: SPEA 2***
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	4.23985	81.4628	0.124055	0.333333	17.4849
2	4.13388	80.1746	0.11682	0.2	17.1988
3	2.41216	83.1411	0.106527	0.111111	20.6213
4	1.68216	89.3874	0.0700265	0.0909091	1.53135
5	0.605503	87.0542	0.0794611	0	12.1146
6	4.47206	78.2228	0.129111	0.5	-21.6284
7	3.17088	86.1351	0.0927838	0.333333	-0.055274
8	0	72.6429	0.149983	0.5	25.4111
9	0.560596	82.9369	0.101825	0	12.6019
10	0.411973	87.2517	0.0926399	0.0833333	11.9123

Runs	D Metric	Execution Time
1	2.80313e-005	6188
2	0.000295998	6250
3	0.000516919	6172
4	0.00051366	6235
5	0.00022422	6187
6	0.000141733	6172
7	0.000155687	6218
8	3.81067e-005	6172
9	0.000290647	6250
10	0.000607382	6172

Table 41: Detail Results (SCH - 12000) SPEA 2*

Test Function	**: Schaffer**
Algorithm	**: SPEA 2***
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	1.15235	86.1171	0.0647858	0.0714286	4.34636
2	0.181473	86.0094	0.0619549	0	10.245
3	4.17356	76.2854	0.122	0.4	-30.4027
4	2.12027	84.5233	0.0734171	0.142857	2.13573
5	1.699	85.9097	0.066907	0.4	-13.6947
6	5.75853	72.6928	0.137502	0.333333	9.21077
7	0.37225	80.5793	0.0974024	0	12.9799
8	0.610456	87.0451	0.0517299	0	9.70435
9	0.10341	81.6498	0.093429	0	12.8986
10	2.52515	82.485	0.0978221	0.222222	-13.9786

Runs	D Metric	Execution Time
1	0.000885309	11328
2	0.00016904	11265
3	0.000102575	11750
4	0.000334209	11250
5	0.000192368	11407
6	6.62638e-005	11063
7	0.000222309	11031
8	0.000247266	10968
9	0.000830338	10907
10	0.000465406	11188

Table 42: Detail Results (SCH - 5000) SPEA 2*

Test Function	**: Schaffer**
Algorithm	**: SPEA 2***
Maximum Evaluations	**: 5000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	2.98092	74.5917	0.0907294	0.333333	-23.5602
2	0	9.38501	0.452532	1	-31.939
3	1.0204	76.5101	0.0845583	0.0833333	12.0409
4	3.65446	71.2795	0.118141	0.5	-22.2403
5	0	0	0.495977	1	-38.1212
6	3.40208	73.8847	0.0958613	0.25	17.2947
7	0	68.7544	0.124789	0.5	24.2354
8	0.343925	79.6865	0.0512265	0	10.4234
9	0.332712	77.8671	0.0668402	0	11.6448
10	0.456301	78.5934	0.0655361	0.142857	9.54765

Runs	D Metric	Execution Time
1	0.000276087	4422
2	0	4359
3	0.000767394	4422
4	0.000141417	4422
5	0	4359
6	0.000257327	4422
7	9.85647e-005	4407
8	0.000637565	4359
9	0.000327574	4438
10	0.000297652	4421

Table 43: Detail Results (SCH - 7000) SPEA 2*

Test Function	**: Schaffer**
Algorithm	**: SPEA 2***
Maximum Evaluations	**: 7000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	4.23985	81.4628	0.124055	0.333333	17.4849
2	4.13388	80.1746	0.11682	0.2	17.1988
3	2.41216	83.1411	0.106527	0.111111	20.6213
4	1.68216	89.3874	0.0700265	0.0909091	1.53135
5	0.605503	87.0542	0.0794611	0	12.1146
6	4.47206	78.2228	0.129111	0.5	-21.6284
7	3.17088	86.1351	0.0927838	0.333333	-0.055274
8	0	72.6429	0.149983	0.5	25.4111
9	0.560596	82.9369	0.101825	0	12.6019
10	0.411973	87.2517	0.0926399	0.0833333	11.9123

Runs	D Metric	Execution Time
1	2.80313e-005	6188
2	0.000295998	6250
3	0.000516919	6172
4	0.00051366	6235
5	0.00022422	6187
6	0.000141733	6172
7	0.000155687	6218
8	3.81067e-005	6172
9	0.000290647	6250
10	0.000607382	6172

Table 44: Detail Results (SCH - 12000) SPEA 2*

Test Function	**: Schaffer**
Algorithm	**: SPEA 2***
Maximum Evaluations	**: 12000**

Runs	Spacing	S Metric	Generational Distance	Coverage	Coverage Difference
1	1.15235	86.1171	0.0647858	0.0714286	4.34636
2	0.181473	86.0094	0.0619549	0	10.245
3	4.17356	76.2854	0.122	0.4	-30.4027
4	2.12027	84.5233	0.0734171	0.142857	2.13573
5	1.699	85.9097	0.066907	0.4	-13.6947
6	5.75853	72.6928	0.137502	0.333333	9.21077
7	0.37225	80.5793	0.0974024	0	12.9799
8	0.610456	87.0451	0.0517299	0	9.70435
9	0.10341	81.6498	0.093429	0	12.8986
10	2.52515	82.485	0.0978221	0.222222	-13.9786

Runs	D Metric	Execution Time
1	0.000885309	11328
2	0.00016904	11265
3	0.000102575	11750
4	0.000334209	11250
5	0.000192368	11407
6	6.62638e-005	11063
7	0.000222309	11031
8	0.000247266	10968
9	0.000830338	10907
10	0.000465406	11188

C. List of Acronyms

DTLZ	Deb Laumanns Thiele Zitzler
EA	Evolutionary Algorithm
FON	Fonseca
KUR	Kursawe
MOEA	Multi Objective Evolutionary Algorithm
MOP	Multi Objective Optimization Problem
M-OPT	Multi-objective Optimization Problems Toolbox
NSGA	Non Dominated Sorting Genetic Algorithm
PAES	Pareto Archived Evolutionary Strategy
POF	Pareto Optimal Front
PSO	Particle Swarm Optimization
SBX	Simulated Binary Crossover
SCH	Schaffer
SPEA	Strength Pareto Evolutionary Algorithm
ZDT	Zitzler Deb Thiele

D. M-OPT Screenshots

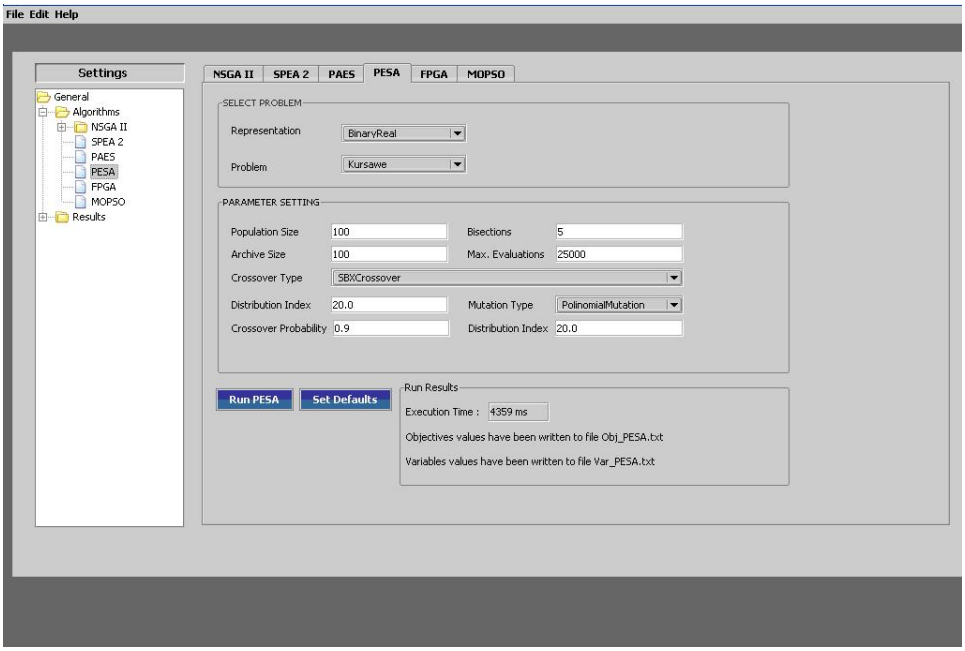


Figure 37: M-OPT Screen1

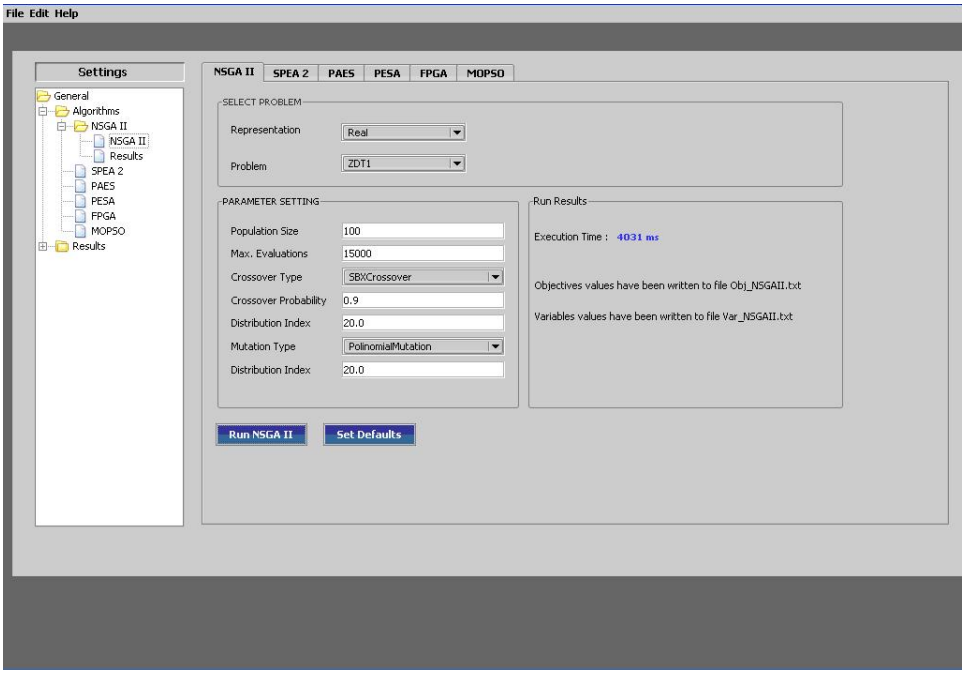


Figure 38: M-OPT Screen2