PDE: A Pareto–frontier Differential Evolution Approach for Multi-objective Optimization Problems

Hussein A. Abbass, Ruhul Sarker, and Charles Newton
School of Computer Science,
University of New South Wales,
University College, ADFA Campus,
Northcott Drive, Canberra ACT, 2600, Australia,
{h.abbass,r.sarker,c.newton}@adfa.edu.au

Abstract-

The use of *evolutionary algorithms* **(EAs) to solve problems with multiple objectives (known as** *Multi-objective Optimization Problems* **(MOPs)) has attracted much attention recently. Being population based approaches, EAs offer a means to find a group of pareto-optimal solutions in a single run.** *Differential Evolution* **(DE) is an EA that was developed to handle optimization problems over continuous domains. The objective of this paper is to introduce a novel** *Pareto–frontier Differential Evolution* **(PDE) algorithm to solve MOPs. The solutions provided by the proposed algorithm for two standard test problems, outperform the** *Strength Pareto Evolutionary Algorithm***, one of the state-of-the-art evolutionary algorithm for solving MOPs.**

1 Introduction

Although single objective decision models are sufficient for some decision making processes, there are many situations where decisions have multiple objectives. Multi-objective problems are known as *multi-objective optimization problems* (MOPs). In these situations, the aim is to simultaneously optimize a group of conflicting objectives. MOPs are a very important research topic, not only because of the multi-objective nature of most real-world decision problems, but also because there are still many open questions in this area. In fact, there is no universally accepted definition of "optimum" in MOP as opposed to single-objective optimization problems, which makes it difficult to even compare results of one method to another. Normally, the decision about what the "best" answer is, corresponds to the so-called human decision maker (Coello 1999).

Traditionally, there are several methods available in the *Operational Research* (OR) literature for solving MOPs as mathematical programming models, viz *goal programming* (Charnes and Cooper 1961), *weighted sum method* (Turban and Meredith 1994), *goals as requirement* (Coello 1999), *goal attainment* (Wilson and Macleod 1993), and the *iso-resource-cost solution* method (Zeleny 1998). The concept of a goal is somewhat different from an objective. A goal is usually considered as a planned objective. Therefore, the optimality is measured, in the case of goal-based methods, in terms of the amount of deviation from the planned levels (*aspiration levels*). Among the previous methods, goal programming is the most widely used in practice although it relies on domain knowledge to setup the goals' aspiration levels. None of the previous methods treat all the objectives simultaneously, except the Iso-resource-cost Solution method, which is a basic requirement in most MOPs. Subsequently, the solutions may be far away from the acceptable ones. These methods handle MOPs with a set of impractical assumptions such as linearity and convexity.

The iso-resource-cost solution method (Zeleny 1998) has been recently demonstrated for a problem with two objectives, two variables and few constraints. To generate the iso-cost solutions, the cost is assumed to equal the total cost of all available resources. Therefore, the set of solutions assumes full utilization of the resource budget. This may lead to many infeasible solutions (under original problem structure) in the solution set (Zeleny 1998). The amount of available resources is decided based on many factors other than the budget, and finding the appropriate mix of resources will make the problem even more complex. However, the concept of iso-resource-cost solutions would be very useful to enhance the future research in MOPs.

In MOPs, there is no single optimal solution, but rather a set of alternative solutions. These solutions are optimal in the wider sense that no other solutions in the search space are superior to (*dominate*) them when all objectives are simultaneously considered. They are known as pareto-optimal solutions. Pareto-optimality is expected to provide flexibility for the human decision maker.

Recently, *evolutionary algorithms* (EAs) were found useful for solving MOPs (Zitzler and Thiele 1999). EAs have some advantages over traditional OR techniques. For example, considerations for convexity, concavity, and/or continuity of functions are not necessary in

EAs, whereas, they form a real concern in traditional OR techniques. Although EAs are successful, to some extent, in solving MOPs, the methods appearing in the literature vary a lot in terms of their solutions and the way of comparing their best results with other existing algorithms. In other words, there is no well-accepted method for MOPs that will produce a good set of solutions for all problems. This motivates the further development of good approaches to MOPs.

In this paper, we develop a novel *Differential Evolution* (DE) algorithm for MOPs. The approach shows promising results when compared with the *Strength Pareto Evolutionary Algorithm* (SPEA) (Zitzler and Thiele 1999), for two benchmark problems. The paper is organized as follows: background materials are scrutinized in Section 2 followed by the proposed algorithm in Section 3. Experiments are then presented in Section 4 and conclusions are drawn in Section 5.

2 Background Materials

2.1 Local and Global optimality in MOPs

Consider a MOP model as presented below:-

$$\text{Optimize } F(\vec{x})$$
$$\text{subject to: } \Omega = \{\vec{x} \in R^n | G(\vec{x}) \leq 0\}$$

Where $\vec{x}$ is a vector of decision variables $(x_1, \ldots, x_n)$ and $F(\vec{x})$ is a vector of objective functions $(f_1(\vec{x}), \ldots, f_K(\vec{x}))$. Here $f_1(\vec{x}), \ldots, f_K(\vec{x})$, are functions on R^n and Ω is a nonempty set in R^n. The vector $G(\vec{x})$ represents constraints that may be easily handled explicitly, such as lower and upper bounds on the variables.

In MOPs, the aim is to find the optimal solution $\vec{x}^* \in \Omega$ which optimize $F(\vec{x})$. Each objective function, $f_i(\vec{x})$, is either maximization or minimization. In this paper, we assume that all objectives are to be minimized for clarity purposes. We may note that any maximization objective can be transformed to a minimization one by multiplying it by -1.

To define the concept of non-dominated solutions in MOPs, we need to define two operators, $\not\geq$ and $\precnsim$ and then assume two vectors, $\vec{x}$ and $\vec{y}$. $\vec{x} \not\geq \vec{y}$ iff $\exists\ x_i \in \vec{x}$ and $y_i \in \vec{y}$ such that $x_i \neq y_i$. And, $\vec{x} \precnsim \vec{y}$ iff $\forall\ x_i \in \vec{x}$ and $y_i \in \vec{y}, x_i \leq y_i$, and $\vec{x} \not\geq \vec{y}$. $\not\geq$ and $\precnsim$ can be seen as the "not equal to" and "less than or equal to" operators respectively, over two vectors. We can now define the concepts of local and global optimality in MOPs.

Definition 1: Neighborhood or open ball The open ball (*ie.* a neighborhood centered on $\vec{x}^*$ and defined by the Euclidean distance) $B_\delta(\vec{x}^*) = \{\vec{x} \in R^n |\ ||\vec{x} - \vec{x}^*|| < \delta\}$.

Definition 2: Local efficient (non-inferior/ pareto-optimal) solution A vector $\vec{x}^* \in \Omega$ is said to be a local efficient solution of MOP iff $\nexists\ \vec{x} \in (B_\delta(\vec{x}^*) \cap \Omega)$ such that $F(\vec{x}) \precnsim F(\vec{x}^*)$ for some positive δ.

Definition 3: Global efficient (non-inferior/ pareto-optimal) solution A vector $\vec{x}^* \in \Omega$ is said to be a global efficient solution of MOP iff $\nexists\ \vec{x} \in \Omega$ such that $F(\vec{x}) \precnsim F(\vec{x}^*)$.

Definition 4: Local non-dominated solution A vector $\vec{y}* \in F(\vec{x})$ is said to be local non-dominated solution of MOP iff its projection onto the decision space, $\vec{x}^*$, is a local efficient solution of MOP.

Definition 5: Global non-dominated solution A vector $\vec{y}* \in F(\vec{x})$ is said to be global non-dominated solution of MOP iff its projection onto the decision space, $\vec{x}^*$, is a global efficient solution of MOP.

In this paper, the term "non-dominated solution" is used as a shortcut for the term "global non-dominated solution".

2.2 MOPs and EAs

EAs for MOPs (Coello 1999) can be categorized as plain aggregating, population-based non-Pareto and Pareto-based approaches. The plain aggregating approaches takes a linear combination of the objectives to form a single objective function (such as in the weighted sum method, goal programming, and goal attainment). This approach produces a single solution at a time that may not satisfy the decision maker, and it requires the quantification of the importance of each objective (*eg.* by setting numerical weights), which is very difficult for most practical situations. However optimizing all the objectives simultaneously and generating a set of alternative solutions, offer more flexibility to decision makers. The simultaneous optimization can fit nicely with population based approaches, such as EAs, because they generate multiple solutions in a single run.

The Vector Evaluated Genetic Algorithm (VEGA) (Schaffer 1985) is a population-based non-Pareto approach. In this approach, the total population is divided into a number of populations equals to the number of objective functions to be optimized. Each population is used to optimize each objective function independently. The populations are then shuffled together followed by conventional crossover and mutation operators. Schaffer (Schaffer 1985) realized that the solutions generated by his system were non-dominated in a local sense, because their non-dominance was limited to the current population, and while a locally dominated individual is also globally dominated, the converse is not necessarily

true.

In the Pareto-based approaches, the dominated and non-dominated solutions in the current population are separated. Goldberg (Goldberg 1989) suggested a non-dominated ranking procedure to decide the fitness of the individuals. Later, Srinivas and Dev (Srinivas and Dev 1994) introduced *Non-dominated Sorting Genetic Algorithms* (NSGA) based on the idea of Goldberg's procedure. The population's individuals are layered according to their ranks. Afterwards, the non-dominated individuals are removed layer by layer from the population.

Fonseca and Fleming (Fonseca and Fleming 1993) proposed a slightly different scheme which is known as *Fonseca and Fleming's evolutionary algorithm* (FFES). In this approach, an individual's rank is determined by the number of individuals dominating it. Without using any non-dominated ranking methods, Horn et al (Horn, Nafpliotis, and Goldberg 1994) proposed the *Niched Pareto Genetic Algorithm* (NPGA) that directly uses a group of randomly picked individuals to form a comparison reference set. The fitness of the two randomly selected individuals is decided according to whether they are dominated by any of the individuals from the comparison reference set. If both individuals are either dominated or non-dominated by the set, then a niched method is used for selection.

The common features of the Pareto-based approaches mentioned above are that (i) the Pareto-optimal solutions in each generation are assigned either the same fitness or a rank, and (ii) some sharing and niche techniques are applied in the selection procedure. Recently, Zitler and Thiele (Zitzler and Thiele 1999) proposed a Pareto-based method, the *Strength Pareto Evolutionary Algorithm* (SPEA). The main features of this approach are: it

1. sorts non-dominated solutions externally and continuously update population,
2. evaluates an individual's fitness depending on the number of external non-dominated points that dominate it,
3. preserves population diversity using the Pareto dominance relationship, and
4. incorporates a clustering procedure in order to reduce the non-dominated set without destroying its characteristics.

Currently, this approach seems to be an outstanding method in the literature.

Most recently, Knowles and Corne (Knowles and Corne 1999; Knowles and Corne 2000) proposed a simple *Evolution Strategies* (ES), (1+1)-ES, known as the *Pareto Archived Evolution Strategy* (PAES) that keeps a record of limited non-dominated individuals. The non-dominated individuals are accepted for recording based on the degree of crowdiness in their grid (defined regions on the Pareto–frontier) location to ensure diversity of individuals in the final solution. The algorithm is strictly confined to local search i.e. it uses a small change (mutation) operator only, and move from a current solution to a nearby neighbor. As they reported, the algorithm works well, specially for problems of low computational complexity. They also propose an extension to this basic approach, which results in some variants of a $(\mu + \lambda)$ -ES. The performance of the algorithm is judged, by solving several test problems, and analyzing the superiority on different regions of the attainment surfaces.

2.3 Statistical Analysis

MOPs require multiple, but uniformly distributed, solutions to form a Pareto trade-off frontier. When comparing two algorithms, these two factors (number of alternative solution points and their distributions) must be considered. There are a number of methods available in the literature to compare the performance of different algorithms. The *error ratio* and the *generational distance* are used as the performance measure indicators when the Pareto optimal solutions are known (Veldhuizen and Mamont 1999). The *spread* measuring technique expresses the distribution of individuals over the non-dominated region (Srinivas and Dev 1994). The method is based on a chi-square-like deviation distribution measure, and it requires several parameters to be estimated before calculating the spread indicator.

The method of *coverage metrics* (Zitzler and Thiele 1999) compares the performances of different multi-objective evolutionary algorithms. It measures whether the outcomes of one algorithm dominate those of another without indicating how much better it is.

Most recently, Knowles and Corne (Knowles and Corne 2000) proposed a method to compare the performances of two or more algorithms by analyzing the distribution of an approximation to the Pareto–frontier. For two objective problems, the *attainment surface* is defined as the lines joining the points on the Pareto–frontier generated by an algorithm. Therefore, for two algorithms A and B, there are two attainment surfaces. A number of sampling lines can be drawn from the origin, which intersects with the attainment surfaces, across the full range of the Pareto–frontier. For a given

sampling line, the intersection of an algorithm closer to the origin (for both minimization) is the winner. Given a collection of k attainment surfaces, some from algorithm A and some from algorithm B, a single sampling line yields k points of intersection, one for each surface. These intersections form a univariate distribution, and we can therefore perform a statistical test to determine whether or not the intersections for one of the algorithms occurs closer to the origin with some statistical significance. Such a test is performed for each of several lines covering the Pareto tradeoff area. Insofar as the lines provide a uniform sampling of the Pareto surface, the result of this analysis yields two numbers - a percentage of the surface in which algorithm A significantly outperforms algorithm B, and the percentage of the surface in which algorithm B significantly outperforms algorithm A.

2.4 Differential Evolution

DE is a branch of evolutionary algorithms developed by Rainer Storn and Kenneth Price (Storn and Price 1995) for optimization problems over continuous domains. In DE, each variable's value in the chromosome is represented by a real number. The approach works by creating a random initial population of potential solutions, where it is guaranteed, by some repair rules, that the value of each variable is within its boundaries. An individual is then selected at random for replacement and three different individuals are selected as parents. One of these three individuals is selected as the main parent. With some probability, each variable in the main parent is changed while at least one variable should be changed. The change is undertaken by adding to the variable's value a ratio of the difference between the two values of this variable in the other two parents. In essence, the main parent's vector is perturbed with the other two parents' vector. This process represents the crossover operator in DE. If the resultant vector is better than the one chosen for replacement, it replaces it; otherwise the chosen vector for replacement is retained in the population. Therefore, DE differs from GA in a number of points:

1. DE uses real number representation while conventional GA uses binary, although it sometimes uses integer or real number representation as well.
2. In GA, two parents are selected for crossover and the child is a recombination of the parents. In DE, three parents are selected for crossover and the child is a perturbation of one of them.
3. The new child in DE replaces a randomly selected vector from the population only if it is better than it. In conventional GA, children replace the parents with some probability regardless of their fitness.

In DE, a solution, l, in generation i is a multi-dimensional vector $\vec{x}^l_{G=i} = (x^l_1, \ldots, x^l_N)^T$. A population, $P_{G=k}$, at generation $G = k$ is a vector of M solutions ($M > 4$). The initial population, $P_{G=0} = \{\vec{x}^1_{G=0}, \ldots, \vec{x}^M_{G=0}\}$, is initialized as

$$x^l_{i,G=0} = lower(x_i) + rand_i[0,1] \times (upper(x_i) - lower(x_i)),$$
$$l = 1, \ldots, M, \quad i = 1, 2, \ldots, N$$

where M is the population size, N is the solution's dimension, and each variable i in a solution vector l in the initial generation $G = 0$, $x^l_{i,G=0}$, is initialized within its boundaries $(lower(x_i), upper(x_i))$. Selection is carried out to select four different solutions indices r_1, r_2, r_3, and $j \in [1, M]$. The values of each variable in the child are changed with some crossover probability, CR, to

$$\forall i \leq N, x'_{i,G=k} = \begin{cases} x^{r_3}_{i,G=k-1} + F \times (x^{r_1}_{i,G=k-1} - x^{r_2}_{i,G=k-1}) \\ \quad if \quad (random[0,1) < CR \wedge i = i_{rand}) \\ x^j_{i,G=k-1} \qquad otherwise \end{cases}$$

where $F \in (0, 1)$ is a problem parameter representing the amount of perturbation added to the main parent. The new solution replaces the old one if it is better than it and at least one of the variables should be changed. The latter is represented in the algorithm by randomly selecting a variable, $i_{rand} \in (1, N)$. After crossover, if one or more of the variables in the new solution are outside their boundaries, the following repair rule is applied

$$x'_{i,G=k} = \begin{cases} \frac{x^j_{i,G} + lower(x_i)}{2} & if \quad x^j_{i,G+1} < lower(x_i) \\ lower(x_i) + \frac{x^j_{i,G} - upper(x_i)}{2} & if \quad x^j_{i,G+1} > upper(x_i) \\ x^j_{i,G+1} & otherwise \end{cases}$$

The DE algorithm is presented in Figure 1.

3 PDE: A Pareto–frontier Differential Evolution algorithm for MOPs

A generic version of the adopted algorithm is presented in Figure 2. The PDE algorithm is similar to the one presented in Figure 1 with the following modifications:-

1. The initial population is initialized according to a Gaussian distribution $N(0.5, 0.15)$.
2. The step-length parameter F is generated from a Gaussian distribution $N(0, 1)$.
3. Reproduction is undertaken only among non-dominated solutions in each generation.

let G denotes a generation, P a population of size M, and $\vec{x}^j_{G=k}$ the j^{th} individual of dimension N in population P in generation k, and CR denotes the crossover probability

input $N, M \geq 4, F \in (0, 1+), CR \in [0, 1]$, *and initial bounds:* $lower(x_i), upper(x_i), i = 1, \ldots N$

initialize $P_{G=0} = \{\vec{x}^1_{G=0}, \ldots, \vec{x}^M_{G=0}\}$ *as*

for each *individual* $j \in P_{G=0}$

$$x^j_{i,G=0} = lower(x_i) + rand_i[0, 1] \times (upper(x_i) - lower(x_i)), i = 1, \ldots, N$$

end for each

evaluate $P_{G=0}$

$k = 1$

while *the stopping criterion is not satisfied* **do**

forall $j \leq M$

randomly select $r_1, r_2, r_3 \in (1, \ldots, M)$, $j \neq r_1 \neq r_2 \neq r_3$

randomly select $i_{rand} \in (1, \ldots, N)$

forall $i \leq N, x'_i =$

$$\begin{cases} x^{r_3}_{i,G=k-1} + F \times (x^{r_1}_{i,G=k-1} - x^{r_2}_{i,G=k-1}) \\ \quad if \quad (random[0, 1) < CR \wedge i = i_{rand}) \\ x^j_{i,G=k-1} \\ \quad otherwise \end{cases}$$

end forall

$$\vec{x}^j_{G=k} = \left\{ \begin{array}{l} \vec{x}' \;\; if \;\; f(\vec{x}') \leq f(\vec{x}^j_{G=k-1}) \\ \vec{x}^j_{G=k-1} \;\; otherwise \end{array} \right\}$$

end forall

evaluate $P_{G=k}$

$k = k + 1$

end while

return *the best encountered solution x.*

Figure 1: The Differential Evolution Algorithm

4. The boundary constraints are preserved either by reversing the sign if the variable is less than 0 or keeping subtracting 1 if it is greater than 1 until the variable is within its boundaries.

5. Offspring are placed into the population if they dominate the main parent.

The algorithm works as follows. An initial population is generated at random from a Gaussian distribution with mean 0.5 and standard deviation 0.15. All dominated solutions are removed from the population. The remaining non-dominated solutions are retained for reproduction. If the number of non-dominated solutions exceeds some threshold, a distance metric relation (will be described in the next paragraph) is used to remove those parents who are very close to each others. Three parents are selected at random. A child is generated from the three parents and is placed into the population if it dominates the first selected parent; otherwise a new selection process takes place.

let G denotes a generation, P a population of size M, and $\vec{x}^j_{G=k}$ the j^{th} individual of dimension N in population P in generation k, and CR denotes the crossover probability

input $N, M \geq 4, \alpha, CR \in [0, 1]$, *and initial bounds:* $lower(x_i), upper(x_i), i = 1, \ldots N$

initialize $P_{G=0} = \{\vec{x}^1_{G=0}, \ldots, \vec{x}^M_{G=0}\}$ *as*

for each *individual* $j \in P_{G=0}$

$$x^j_{i,G=0} = Gussian(0.5, 0.15), i = 1, \ldots, N$$

Repair $\vec{x}^j_{G=k}$ *if any variable is outside its boundaries*

end for each

evaluate $P_{G=0}$

$k = 1$

while *the stopping criterion is not satisfied* **do**

remove all dominated solutions from $P_{G=k-1}$

if *the number of non-dominated solutions in* $P_{G=k-1} > \alpha$, **then** *apply the neighborhood rule*

end if

for $j = 0$ *to number of non-dominated solutions in* $P_{G=k-1}$

$\vec{x}^j_{G=k} \leftarrow \vec{x}^j_{G=k-1}$

end for

while $j \leq M$

randomly select $r_1, r_2, r_3 \in (1, \ldots, \alpha)$, from the *non-dominated solutions of* $P_{G=k-1}$, *where* $r_1 \neq r_2 \neq r_3$

randomly select $i_{rand} \in (1, \ldots, N)$

forall $i \leq N, x'_{i,G=k} =$

$$\begin{cases} x^{r_3}_{i,G=k-1} + Gaussian(0, 1) \times (x^{r_1}_{i,G=k-1} - x^{r_2}_{i,G=k-1}) \\ \quad if \quad (random[0, 1) < CR \wedge i = i_{rand}) \\ x^j_{i,G=k-1} \\ \quad otherwise \end{cases}$$

end forall

Repair $\vec{x}^j_{G=k}$ *if any variable is outside its boundaries*

if $\vec{x}'$ *dominates* $\vec{x}^{r_3}_{G=k-1}$ **then**

$\vec{x}^j_{G=k} \leftarrow \vec{x}'$

$j = j + 1$

end if

end while

$k = k + 1$

end while

return *the set of non-dominated solutions.*

Figure 2: The Pareto–frontier Differential Evolution Algorithm (PDE)

This process continues until the population is completed.

A maximum number of non-dominated solutions in each generation was set to 50. If this maximum is exceeded, the following nearest neighbor distance function is adopted:

$$D(x) = \frac{(min||x - x_i|| + min||x - x_j||)}{2},$$

where $x \neq x_i \neq x_j$. That is, the nearest neighbor distance is the average Euclidean distance between the

closest two points. The non-dominated solution with the smallest neighbor distance is removed from the population until the total number of non-dominated solutions is retained to 50.

4 Experiments

4.1 Test Problems

The algorithm is tested on the following two benchmark problems used in Zitler and Thiele (1999):

Test Problem 1: Convex

$$f_1(x) = x_1$$

$$f_2(x) = g \times (1 - \sqrt{(\frac{f_1}{g})})$$

$$g = 1 + 9 \times \frac{(\sum_{i=2}^{n} x_i)}{(n-1)}$$

$$x_i \in [0,1], i = 1, \ldots, 30$$

Test Problem 2: Discontinuous pareto–front

$$f_1(x) = x_1$$

$$f_2(x) = g * (1 - \sqrt{\frac{f_1}{g}} - (\frac{f_1}{g}) \sin(10\pi f_1))$$

$$g = 1 + 9 \times \frac{(\sum_{i=2}^{n} x_i)}{(n-1)}$$

$$x_i \in [0,1], i = 1, \ldots, 30$$

Both test problems contain two objective functions and thirty variables. The computational results of these test problems are provided in the next section.

4.2 Experimental Setup

The initial population size is set to 100 and the maximum number of generations to 200. Twenty different crossover rates changing from 0 to 1.00 with an increment of 0.05 are tested without mutation. The initial population is initialized according to a Gaussian distribution $N(0.5, 0.15)$. Therefore, with high probability, the Gaussian distribution will generate values between $0.5 \pm 3 \times 0.15$ which fits with the variables' boundaries. If a variable's value is not within its range, a repair rule is used to repair the boundary constraints. The repair rule is simply to truncate the constant part of the value; therefore if, for example, the value is 3.3, the repaired value will be 0.3 assuming that the variable is between 0 and 1. The step-length parameter F is generated for each variable from a Gaussian distribution $N(0, 1)$. The algorithm is written in standard C^{++} and ran on a Sun Sparc 4.

4.3 Experimental Results and Discussions

In Figure 3, we plotted all the non-dominated solutions for the first twenty runs of both test problems with the SPEA results obtained from the web site "http//www.tik.ee.ethz.ch/∼zitzler/testdata.html". The crossover rates of the solutions plotted were 0.15 and 0.05 for the first and second test problems respectively. As can be seen in Figure 3, our results are clearly better than SPEA in terms of the objective function's values. The Pareto–frontier is always lower than SPEA and the distribution of the points on the Pareto–frontier is more uniformly distributed than SPEA.

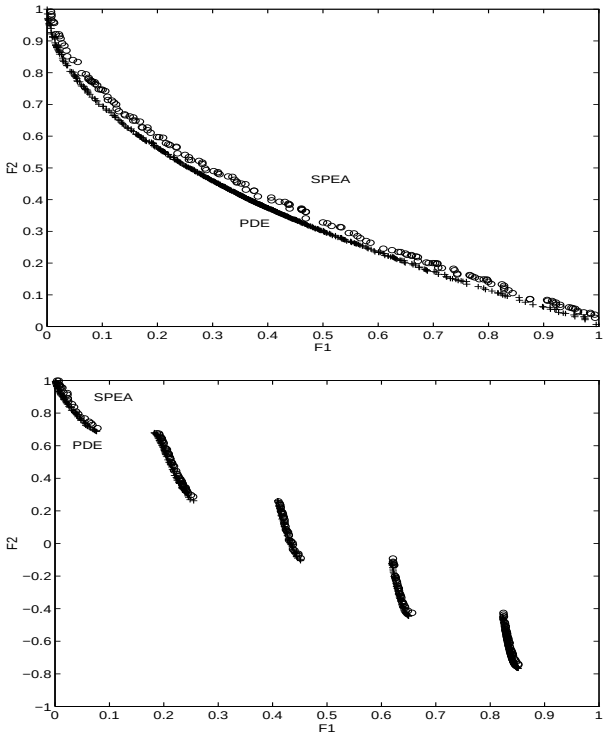

Figure 3: The performance of the PDE algorithm compared with SPEA on the test problem.

To perform the statistical analysis using Knowles and Corne method (Knowles and Corne 2000), we used the solutions of the twenty runs for each crossover rate. The results of the comparison is presented in the form of a pair [a,b], where a gives the percentage of the space

(i.e. the percentage of lines) on which algorithm A is found statistically superior to B, and b gives the similar percentage for algorithm B. For problem1, the best result [84.3,15.1] is achieved with crossover rate 0.15. This means, our algorithm outperforms SPEA on about 84.3 percent of the Pareto surface whereas SPEA is statistically superior than our algorithm for 15.1 percent. For problem2, the best result is obtained with crossover 0.05.

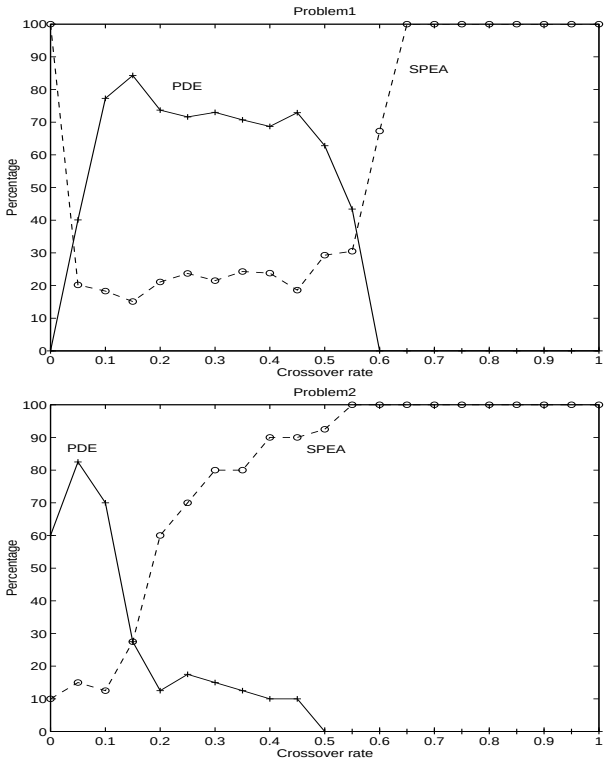

Figure 4: The percentage outperformed by our algorithm and SPEA for the two test problems. The x-axis represents the crossover rate for our algorithm and the y-axis represents the percentage outperformed by both algorithms.

As per the analysis, the percentage outperformed by our algorithm and SPEA are plotted against the crossover rate in Figure 4 for both test problems. For SPEA, the results are the best published results; therefore, the crossover rate on the x-axis does not reflect the crossover rate used in SPEA. Only within the crossover range 0.05 - 0.55 for problem1 and 0.0 - 0.15 for problem2, PDE is significantly better than SPEA. The crossover rate versus the number of non-dominated solution points are shown in Figure 5. In both problems, the number of solution points are maximum within the crossover range 0.10 to 0.30. Interestingly, the distribution of non-dominated solutions against the crossover rate follows a normal distribution shape.

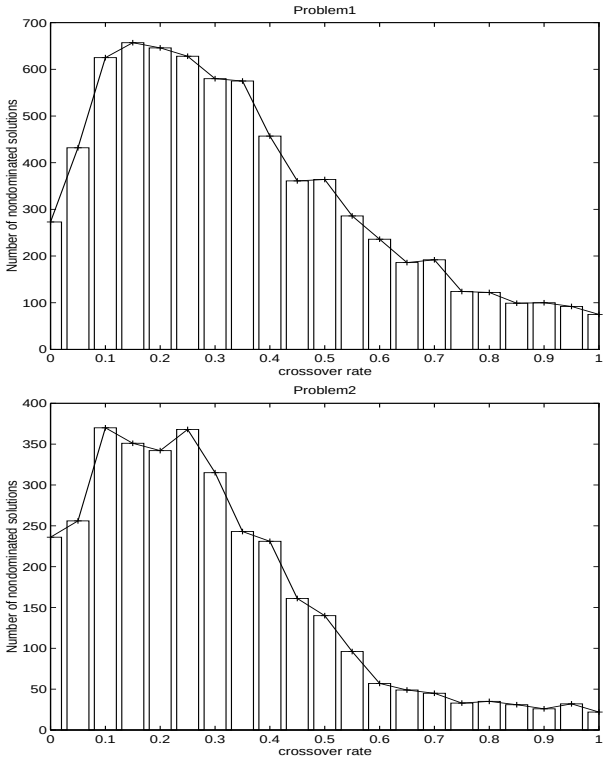

Figure 5: The distribution of the number of non-dominated solutions found by our algorithm for the two test problems using different crossover rates. The x-axis represents the crossover rate and the y-axis represents the number of non-dominated solutions found.

From the experimental results, it is clear that the solution's quality varies with the crossover rate. However, the results suggest that there is a trend in both problems which may suggest that the relationship between the crossover rate and the solution's quality is almost unimodal. This is very interesting sine it makes the search problem of finding a good crossover rate easy.

5 Conclusions and Future Research

In this paper, a novel differential evolution approach is presented for vector optimization problems. The approach generates a step by mutation, where the step is randomly generated from a Gaussian distribution. We tested the approach on two benchmark problems and it was found that our approach outperformed the SPEA approach. We also experimented with different crossover and mutation rates, on these two test problems, to find their best solutions. The crossover rates are found to be very sensitive to the solutions. However, a trend was found which suggests that large number of non-dominated solutions were found with low-crossover rates.

For future work, we intend to test the algorithm on more problems. Also, the parameters chosen in this paper were generated experimentally. It would be interesting to see the effect of these parameters on the problem.

Bibliography

Charnes, A. and W. Cooper (1961). *Management models and industrial applications of linear programming, volume 1*. John Wiley, New York.

Coello, C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems 1*(3), 269–308.

Fonseca, C. and P. Fleming (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California*, 416–423.

Goldberg, D. (1989). *Genetic algorithms: in search, optimisation and machine learning*. Addison Wesely.

Horn, J., N. Nafpliotis, and D. Goldberg (1994). A niched pareto genetic algorithm for multiobjective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation 1*, 82–87.

Knowles, J. and D. Corne (1999). The pareto archived evolution strategy: a new baseline algorithm for multiobjective optimization. *In 1999 Congress on Evolutionary Computation, Washington D.C., IEEE Service Centre*, 98–105.

Knowles, J. and D. Corne (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation 8*(2), 149–172.

Schaffer, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, 93–100.

Srinivas, N. and K. Dev (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation 2*(3), 221–248.

Storn, R. and K. Price (1995). Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley.

Turban, E. and J. Meredith (1994). *Fundamentals of Management Science*. McGraw-Hill, Boston, USA.

Veldhuizen, D. V. and G. Mamont (1999). Multiobjective evolutionary algorithm test suites. *Procedings of the 1999 ACM Sysposium on Applied Computing, San Antonio, Texas, ACM*, 351–357.

Wilson, P. and M. Macleod (1993). Low implementation cost iir digital filter design using genetic algorithms. *IEE/IEEE workshop on Natural Algorithms in Signal Processing*, 1–8.

Zeleny, M. (1998). Multiple criteria decision making: Eight concepts of optimality. *Human Systems Management 17*, 97–107.

Zitzler, E. and L. Thiele (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation 3*(4), 257–271.