Solving Multiobjective Optimization Problems using Evolutionary Algorithm

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

Abstract

Being capable of finding a set of pareto–optimal solutions in a single run, which is a necessary feature for multi–criteria decision making, Evolutionary Algorithms (EAs) has attracted many researchers and practitioners to address the solution of Multiobjective Optimization Problems (MOPs). In a previous work, we developed a Pareto Differential Evolution (PDE) algorithm to handle multiobjective optimization problems. Despite the overwhelming number of Multiobjective Evolutionary Algorithms (MEAs) in the literature, little work has been done to identify the best MEA using an appropriate assessment methodology. In this paper, we compare our algorithm with twelve other well-known MEAs, using a popular assessment methodology, by solving two bench-mark problems. The comparison shows the superiority of our algorithm over others.

1 Introduction

Multiobjective Optimization Problems (MOPs) optimize a set of conflicting objectives simultaneously. 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 one 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 (Coello 1999). None of the OR methods treat all the objectives simultaneously which is a basic requirement in most MOPs. In addition, these methods handle MOPs with a set of impractical assumptions such as linearity and convexity.

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

Recently, *evolutionary algorithms* (EAs) were found to be 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.

Recently, we developed a novel *Differential Evolution* (DE) algorithm for MOPs (Abbass, Sarker, and Newton 2001). The approach showed promising results when compared with the *Strength Pareto Evolutionary Algorithm* (SPEA) (Zitzler and Thiele 1999), for two benchmark problems. However there are several other known methods such as Fonseca and Fleming's genetic algorithm (FFGA) (Fonseca and Fleming 1993), Hajela's and Lin's genetic algorithm (HLGA)(Hajela and Lin 1992), Niched Pareto Genetic Algorithm (NPGA) (Horn, Nafpliotis, and Goldberg 1994), Non-dominated Sorting Genetic Algorithms (NSGA)(Srinivas and Dev 1994), Random Sampling Algorithm (RAND)(Zitzler and Thiele 1999), Single Objective Evolutionary Algorithm (SOEA) (Zitzler and Thiele 1999), Vector Evaluated Genetic Algorithm (VEGA) (Schaffer 1985) and Pareto Archived Evolution Strategy (PAES) (Knowles and Corne 1999; Knowles and Corne 2000). There are several versions of PAES like PAES, PAES20, PAES98 and PAES98mut3p. In this paper, we compare the solutions of two benchmark problems, produced by our DE algorithm with all these methods, using a statistical comparison technique recently proposed by Knowles and Corne (Knowles and Corne 1999; Knowles and Corne 2000). From the comparison, it is clear that our algorithm outperforms most algorithms when applied to these two test problems.

The paper is organized as follows. After introducing the research context, previous research is 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 Previous Research

2.1 Existing MEAs

MEAs for solving MOPs can be categorized as plain aggregating, population-based non-Pareto and Pareto-based approaches (Coello 1999). In this section, we would briefly discuss several population-based approaches as they are more successful when solving MOPs, and are popular among researchers and practitioners.

The Random Sampling Evolutionary Algorithm (RAND)(Zitzler and Thiele 1999) generates randomly a certain number of individuals per generation, according to the rate of crossover and mutation (though neither crossover, mutation nor selection are performed). Hence the number of fitness evaluations was the same as for the EAs. Another algorithm called Single Objective Evolutionary Algorithm (SOEA)(Zitzler and Thiele 1999) uses the weighted-sum aggregation. In contrast to other algorithms, 100 independent runs were performed per test problem, each run being optimized towards another randomly chosen linear combination of the objectives. The nondominated solutions among all solutions generated in the 100 runs form the trade-off frontier achieved on a particular test problem.

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 equal 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.

Hajela's and Lin's genetic algorithm (HLGA)(Hajela and Lin 1992) is also a non-Pareto approach that uses the weighted-sum method for fitness assignment. Thereby, each objective is assigned a

weight between zero and one, with the sum of all weights being exactly equal to one. To search for multiple solutions in parallel, the weights are encoded in the genotype. The diversity of the weight combinations is promoted by phenotypic fitness sharing. As a consequence, the EA evolves solutions and weight combinations simultaneously.

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. In this method, the fitness assignment is carried out through several steps. In each step, the nondominated solutions constituting a nondominated frontier are assigned the same dummy fitness value. These solutions have the same fitness values and are ignored in the further classification process. Finally, the dummy fitness is set to a value less than the smallest shared fitness value in the current nondominated frontier. Then the next frontier is extracted. This procedure is repeated until all individuals in the population are classified.

Fonseca and Fleming (Fonseca and Fleming 1993) proposed a slightly different scheme which is known as Fonseca and Fleming's genetic algorithm (FFGA). 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 combines tournament selection and the concept of Pareto dominance. Two competing individuals and a comparison set of other individuals are picked at random from the population; the size of the comparison set is given by a user defined parameter. If one of the competing individuals is dominated by any member of the set and the other is not, then the later is chosen as the winner of the tournament. If *both* individuals are dominated (or not dominated), the result of the tournament is decided by sharing: the individual that has the least individuals in its niche (defined by the *niche radius*) is selected for reproduction. Horn and Nafpliotis (Horn, Nafpliotis, and Goldberg 1994) used phenotypic sharing on the objective vectors.

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 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 (i) sorts the non-dominated solutions externally and continuously updates the population, (ii) evaluates an individual's fitness depending on the number of external non-dominated points that dominate it, (iii) preserves population diversity using the Pareto dominance relationship, and (iv) incorporates a clustering procedure in order to reduce the non-dominated set without destroying its characteristics.

Most recently, Knowles and Corne (Knowles and Corne 1999; Knowles and Corne 2000) proposed a simple Evolution Strategy (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. They also proposed an extension to this basic approach, which results in some variants of a $(\mu + \lambda)$ -ES. These are recognized as PAES (on-line performance using an archive of 100 solutions), PAES20 (off-line performance using an archive of 20 solutions), PAES98 (off-line performance using an archive of 98 solutions) and PAES98mut3p (PAES98 but with a mutation rate of 3 percent).

Our algorithm is a Pareto-based approach using Differential Evolution (DE) for multi-objective optimization (Abbass, Sarker, and Newton 2001). This algorithm is briefly introduced in a later

section.

2.2 Comparison Techniques

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 in addition to the quality of solutions. There are a number of assessment methodologies 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 by indicating whether the outcomes of one algorithm dominate those of another without measuring how much better it is.

A statistical comparison method called "attainment surfaces" was introduced by Fonseca & Fleming (1996). For two objective problems, the *attainment surface* is defined as the lines joining the points (candidate solutions) on the Pareto–frontier generated by an algorithm. Therefore, for two algorithms A and B, there are two attainment surfaces. An attainment surface divides the objective space into two regions: one containing vectors which are dominated by the results produced by the algorithm, and another that contains vectors that dominate the results produced by the algorithm. 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. Fonseca and Fleming's idea was to consider a collection of sampling lines which intersect the attainment surfaces across the full range of the Pareto frontier.

If MEAs are run r times, each algorithm will return r attainment surfaces, one from each run. Having these r attainment surfaces, some from algorithm A and some from algorithm B, a *single sampling line* yields r points of intersection, one for each surface. These intersections form a univariate distribution, and therefore, we can perform standard non-parametric statistical procedures to determine whether or not the intersections for one of the algorithms occurs closer to the origin with some statistical significance. Such statistical tests have been performed by (Knowles and Corne 2000) for each of the lines covering the Pareto trade–off 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 outperforms algorithm B with statistical significance, and that when algorithm B outperforms algorithm A.

Knowles and Corne (2000) presented their results of a comparison 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 was found statistically superior to B, and b gives the similar percentage for algorithm B. Typically, if both A and B are 'good', then $a + b < 100$. The quantity [100 - (a + b)], of course, gives the percentage of the space on which the results were statistically inconclusive. They use statistical significance at the 95 percent confidence level. Knowles and Corne (2000) also extended their comparison methodology to comparing more than two algorithms.

If the algorithms are competitive, the results of the statistical test may vary with the number of sampling lines drawn since the procedure considers only the intersection points of sampling lines

and attainment surfaces. Knowles and Corne (2000) proposed that 100 lines should be adequate, although, obviously, more lines the better. They have shown experimentally that the percentage of the space $(a + b)$ increases, to give statistically significant results, with the increased number of lines.

2.3 Differential Evolution

DE is a branch of evolutionary algorithms developed by Storn and 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 but 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 by the other two parents' vectors. 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 GA 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 k is a multi-dimensional vector $\vec{x}^l_{G=k} = (x^l_1, \ldots, x^l_N)^T$. 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

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

where M is the population size, N is the solution's dimension, and each variable x_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

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

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 until the boundary constraints are satisfied

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

3 A Differential Evolution algorithm for MOPs

A generic version of the adopted algorithm is presented in Figure 3 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.
4. Offspring are placed into the population if they dominate the main parent.
5. The boundary constraints are preserved either by reversing the sign if the variable is less than 0 or keep subtracting 1 if it is greater than 1 until the variable is within its boundaries.

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 (Abbass, Sarker, and Newton 2001) is used to remove those parents who are very close to each other. 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. 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 neighborhood 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 neighborhood distance is the average Euclidean distance between the closest two points. The non-dominated solution with the smallest neighborhood distance is removed from the population until the total number of non-dominated solutions is retained at 50.

4 Experiments

4.1 Test Problems

The algorithm is tested using the following two benchmark problems from Zitler and Thiele (1999):

Test Problem 1: Convex functions

$$f_1(x) = x_1$$

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

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

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

Test Problem 2: Discontinuous functions

$$f_1(x) = x_1$$

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

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

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

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

For our algorithm, 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 boundary, a repair rule is used to repair the boundary constraints. The repair rule is applied 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 this section, the solutions of two test problems, provided by our PDE algorithm, are compared with the solutions of twelve other MEAs (FFGA, HLGA, NPGA, NSGA, RAND, SOEA, SPEA, VEGA, PAES, PAES20, PAES98 and PAES98mut3p) using a statistical comparison technique. The results of other algorithms, except PAESs, were obtained from the web site "http//www.tik.ee.ethz.ch/∼zitzler/testdata.html". The results for all PAESs were obtained from "http://www.rdg.ac.uk/∼ssr97jdk/multi/PAES.html".

To perform the statistical analysis using the Knowles and Corne method (Knowles and Corne 2000), we used the solutions of twenty runs of the DE algorithm for each crossover rate. The results of the comparison is presented in the form of a pair [a,b] for each crossover rate, where a gives the percentage of the space (i.e. the percentage of lines) on which PDE algorithm is found statistically superior to the other, and b gives the similar percentage for the other algorithm. For example, for test problem 1, the best result using PDE [84.3,15.1] is achieved with crossover rate 0.15 when

compared to SPEA. 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 problem 2, the best result is obtained with crossover 0.05 when compared to SPEA.

In Figures 1 and 2, the x-axis represents the crossover rate used in our PDE algorithm and the y-axis is the percentage of superiority. Each figure contains a plot of "a" for our PDE algorithm and "b" for one of the other existing algorithm for a given problem. Twelve plots in Figure 1 show the comparison of PDE with each of the others MEAs for test problem 1, and Figure 2 shows the same for test problem 2.

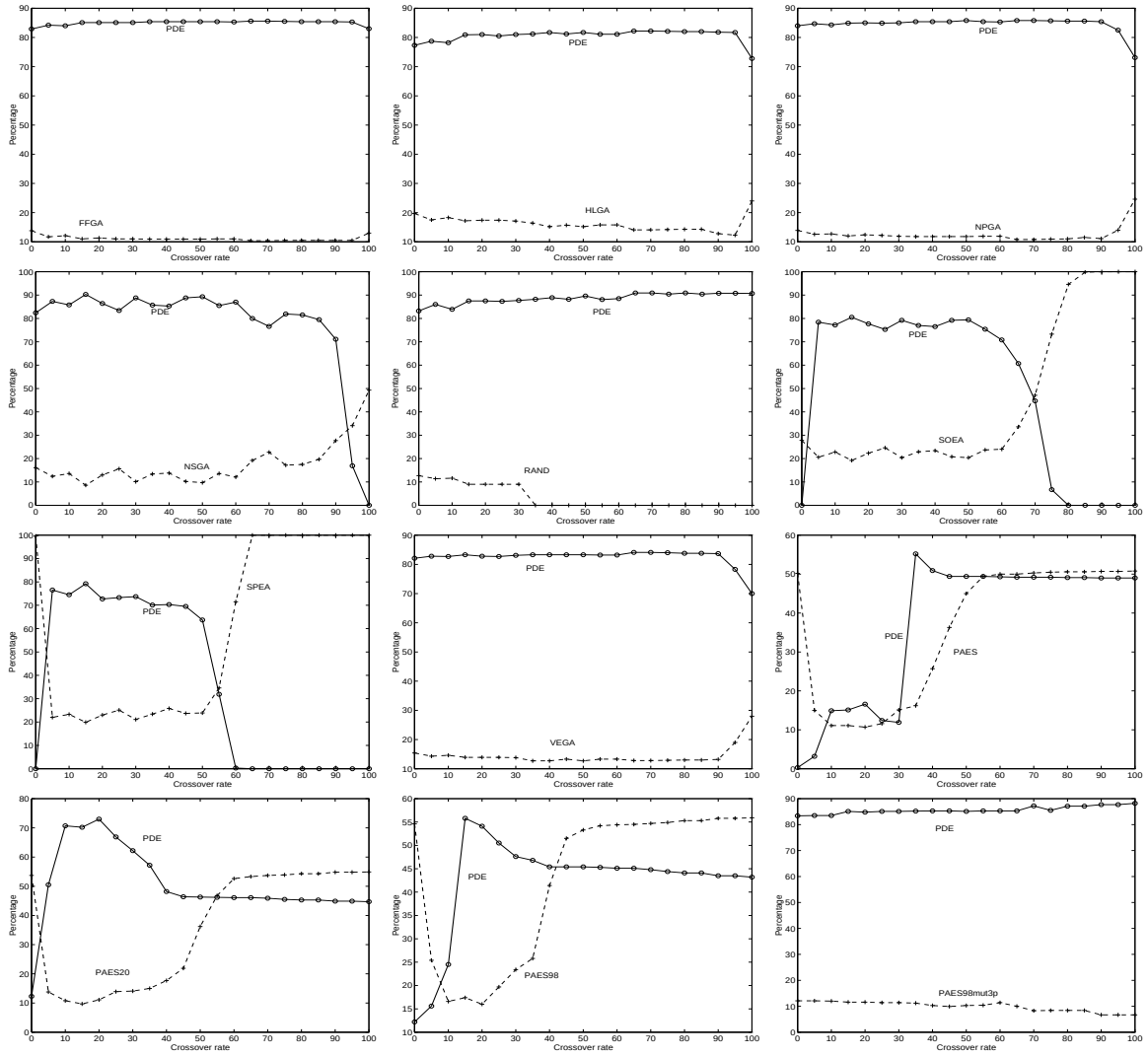

Figure 1: Test Problem 1

For both test problems, PDE is significantly better than FFGA, HLGA, NPGA, Rand and VEGA irrespective of the crossover rate. PDE is much better than NSGA for any crossover rate less than 0.85 for problem 1 and 0.8 for problem 2. PDE is superior than SOEA within the crossover rate 0.05 to 0.65 and SPEA within 0.05 to 0.5 for test problem 1. These figures for test problem 2 are 0 to 0.45 and 0.05 to 0.1 respectively. PDE is clearly better than PAES, PAES98 and PAES98mut3p for both test problems within certain range of crossover rate. Although PDE shows superiority over PAES20 for test problem 1, it shows very little success for test problem 2. For test problem

1, a range of crossover rate for PDE can successfully challenge all other MEAs. For example, the solutions of PDE at a crossover rate of 0.35 outperforms all other algorithms. From these results, it can be stated that no algorithm (out of 12) produces optimal solutions. However, PDE solutions could be close to the pareto frontier though there is no guarantee. For problem 2, there is no single crossover rate for which PDE is superior than all the other MEAs. However such a rate can be found when we exclude one or two MEAs. That means, no one is close to optimal although PDE outperforms most algorithms.

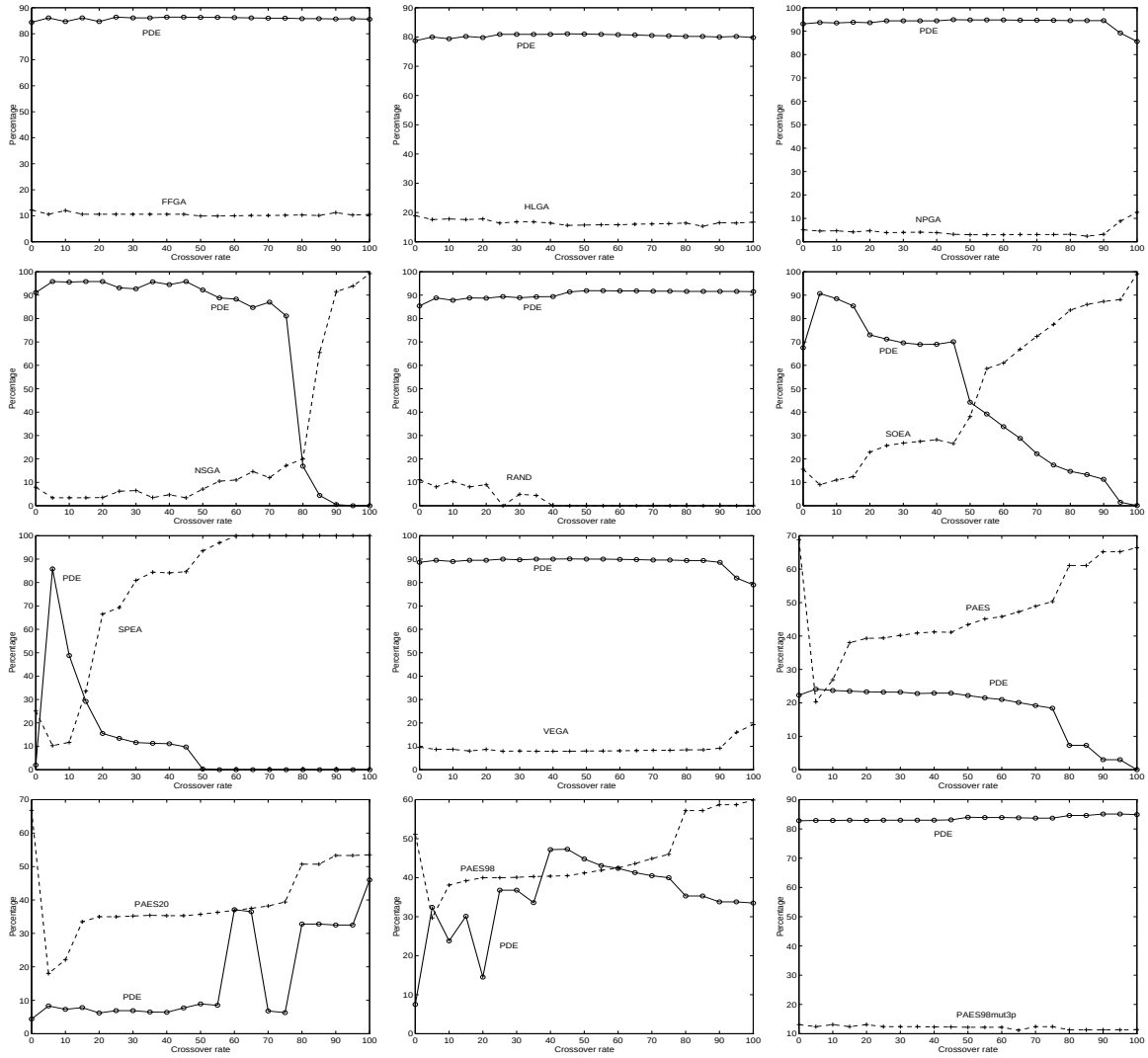

Figure 2: Test Problem 2

5 Conclusions and Future Research

In this paper, a novel differential evolution approach is discussed for multiobjective 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 almost all existing MEAs. 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 sensitive when compared with certain MEAs. However, a trend was found which suggests that a large number of non-dominated solutions were found with low-crossover rates. In future work, we intend to test the algorithm on more problems.

Bibliography

Abbass, H., R. Sarker, and C. Newton (2001). A pareto differential evolution approach to vector optimisation problems. *Congress on Evolutionary Computation 2*, 971–978.

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.

Hajela, P. and C. Lin (1992). Genetic search strategies in multicriterion optimal design. *Structural Optimization 4*, 99–107.

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.

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.

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.

Appendix: The Pareto Differential Evolution Algorithm

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* $j \in P_{G=0}$

    $x^{j}_{i,G=0} = Gaussian(0.5, 0.15), i = 1, \ldots, N$

    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*

  for $j = 0$ *to number of non-dominated solutions in* $P_{G=k-1}$ $\vec{x}^{j}_{G=k} \leftarrow \vec{x}^{j}_{G=k-1}$

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

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

    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 3: The Pareto–frontier Differential Evolution Algorithm (PDE)