International Journal on Artificial Intelligence Tools

(Leave $1\frac{1}{2}$ inch blank space for Publisher)

The Pareto Differential Evolution Algorithm

H. A. Abbass, and R. Sarker
School of Computer Science,University of New South Wales,University College, ADFA Campus,Northcott Drive, Canberra ACT, 2600, Australia
{*h.abbass,r.sarker*}*@adfa.edu.au*

Received (received date)
Revised (revised date)

Abstract

The use of evolutionary algorithms (EAs) to solve problems with multiple objectives (known as *Vector Optimization Problems* (VOPs)) 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 Differential Evolution* (PDE) algorithm to solve VOPs. The solutions provided by the proposed algorithm for five standard test problems, is competitive to nine known evolutionary multiobjective algorithms for solving VOPs.

Keywords: Differential Evolution, Evolutionary Multiobjective

1. Introduction

Although single objective decision models are sufficient for some decision making processes, there are many situations where problems involve multiple objectives. Multi–objective problems are known as *Vector optimization problems* (VOPs). In these situations, the aim is to simultaneously optimize a group of conflicting objectives. VOPs 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 VOP 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[2].

Traditionally, there are several methods available in the *Operational Research* (OR) literature for solving VOPs as mathematical programming models, viz *goal programming* [1], *weighted sum method*[13], *goals as requirement*[2], *goal attainment* [15], and the *iso–resource–cost solution* method [16]. 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 traditional methods treat all the objectives simultaneously, except the Iso–resource–cost Solution method, which is a basic requirement in most VOPs. Subsequently, the solutions may be far away from the acceptable ones. These methods handle VOPs with a set of impractical assumptions such as linearity and convexity.

The iso–resource–cost solution method[16] 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[16]. 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 VOPs.

In VOPs, 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 in multiobjective optimization.

Recently, *evolutionary algorithms* (EAs) were found useful for solving VOPs[17]. 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 VOPs, 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 VOPs that will produce a

good set of solutions for all problems. This motivates the further development of good approaches to VOPs.

In this paper, we propose a novel *Differential Evolution* (DE) algorithm for VOPs. The approach shows promising results when compared with nine evolutionary multiobjective algorithms for five benchmark problems. The motivation for the current study was initially to use an efficient evolutionary multiobjective algorithm for training artificial neural networks. This dragged us into DE as an evolutionary algorithm which is designed for continuous domains and use quite different crossover operator from other EAs. We then developed the PDE algorithm introduced in this paper and decided to start testing it on conventional optimization problems before using it for artificial neural networks. At the moment, we are using PDE for prediction and classification by neural networks and it is found to be very competitive against artificial neural networks training's approaches.

The paper is organized as follows: background materials are scrutinized in Section followed by the proposed algorithm in Section . Experiments are then presented in Section and conclusions are drawn in Section .

2. Background Materials

2.1. *Local and Global optimality in VOPs*

Consider a VOP model as presented below:-

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

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 VOPs, 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 VOPs, we need to

define two operators, $\not\succeq$ and $\precnsim$ and then assume two vectors, $\vec{x}$ and $\vec{y}$. $\vec{x} \not\succeq \vec{y}$ 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 $\vec{x} \not\succeq \vec{y}$. $\not\succeq$ 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 VOPs.

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 VOP 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 VOP 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 VOP iff its projection onto the decision space, $\vec{x}^*$, is a local efficient solution of VOP.

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

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

2.2. *VOPs and EAs*

EAs for VOPs[2] 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 Random Sampling Evolutionary Algorithm (RAND)[17] 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) [17] 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)[10] 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[10] 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 [5] suggested a non–dominated ranking procedure to decide the fitness of the individuals. Later, Srinivas and Dev [11] 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.

Hajela's and Lin's genetic algorithm (HLGA)[6] 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 [5] suggested a non-dominated ranking procedure to decide the fitness of the individuals. Later, Srinivas and Dev [11] 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[4] 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[7] 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[7] 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 a rank, and (ii) some sharing and niche techniques are applied in the selection procedure. Recently, Zitler and Thiele[17] 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.

Most recently, Knowles and Corne [8,9] proposed a simple Evolution Strategies, (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*

VOPs 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 [14]. The *spread* measuring technique expresses the distribution of individuals over the non–dominated region [11]. 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*[17] 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 [9] 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*

Evolutionary algorithms[3] is a kind of global optimization techniques that use selection and recombination as their primary operators to tackle optimization problems. *Differential evolution* (DE) is a branch of evolutionary algorithms developed by Rainer Storn and Kenneth Price [12] for optimization problems over continuous domains. In DE, each variable is represented in the chromosome by a real number. The approach works as follows:-

1. Create an initial population of potential solutions at random, where it is guaranteed, by some repair rules, that variables' values are within their boundaries.
2. Until termination conditions are satisfied
   (a) Select at random a trail individual for replacement, an individual as the main parent, and two individuals as supporting parents.
   (b) With some probability, called the *crossover probability*, each variable in the main parent is perturbed by adding to it a ratio, F, of the difference between the two values of this variable in the other two supporting parents. At least one variable must be changed. This process represents the crossover operator in DE.
   (c) If the resultant vector is better than the trial solution, it replaces it; otherwise the trial solution is retained in the population.
   (d) go to 2 above.

From the previous discussion, DE differs from *genetic algorithms* (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.

3. PDE: The Pareto Differential Evolution algorithm for VOPs

The Pareto–frontier Differential Evolution (PDE) algorithm for vector optimization problems is an adaptation of the DE algorithm described in the previous section 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. 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. Assuming that all variables are bounded between (0,1), 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. Three parents are selected at random (one as a main and also trial solution and the other two as supporting parents). A child is generated from the three parents and is placed into the population if it dominates the main parent; otherwise a new selection process takes place. This process continues until the population is completed. A generic version of the adopted algorithm follows

1. Create a random initial population of potential solutions. Each variable is assigned a random value according to a Gaussian distribution $N(0.5, 0.15)$.
2. Repeat
   (a) Evaluate the individuals in the population and label those who are non–dominated.
   (b) If the number of non–dominated individuals in the population is less than 3 repeat the following until the number of non–dominated individuals in the population is greater than or equal to 3.
      i. Find a non–dominated solution among those who are not labelled.

ii. Label the solution as non–dominated.

(c) If the number of non–dominated individuals in the population is greater than the allowed maximum, apply the neighborhood distance function until the number of non–dominated individuals in the population is less than the allowed maximum.

(d) Delete all dominated solutions from the population.

(e) Repeat

i. Select at random an individual as the main parent α_1, and two individuals, α_2, α_3 as supporting parents.

ii. Select at random a variable j.

iii. For each variable i

A. With some probability $Uniform(0,1)$ or if $i = j$, do

$$x_i^{child} \leftarrow x_i^{\alpha_1} + F(x_i^{\alpha_2} - x_i^{\alpha_3}) \tag{1}$$

otherwise

$$x_i^{child} \leftarrow x_i^{\alpha_1} \tag{2}$$

where each variable i in the main parent, $x_i^{\alpha_1}$, is perturbed by adding to it a ratio, $F \in Gaussian(0,1)$, of the difference between the two values of this variable in the two supporting parents. At least one variable must be changed.

iv. If the child dominates the main parent, place it into the population.

(f) Until the population size is M

3. Until termination conditions are satisfied, go to 2 above.

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.

We may note here that similar to the conventional DE algorithm and on the contrary to other evolutionary algorithms, the algorithm requires in step (ii) of (d) above that at least one variable be crossovered. Therefore, even if the crossover probability is zero, PDE still performs crossover (Step A of e–iii).

4. Experiments

4.1. *Test Problems*

The algorithm is tested on the following five 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: Non–convex counterpart to test problem 1

$$f_1(x) = x_1$$
$$f_2(x) = g \times (1 - (\frac{f_1}{g}))^2$$
$$g = 1 + 9 \times \frac{\sum_{i=2}^{n} x_i}{n-1}$$
$$x_i \in [0,1], i = 1, \ldots, 30$$

Test Problem 3: Discontinuous pareto–front

$$f_1(x) = x_1$$
$$f_2(x) = g * (1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\sin(10\pi f_1))$$
$$g = 1 + 9 \times \frac{(\sum_{i=2}^{n} x_i)}{(n-1)}$$
$$x_i \in [0,1], i = 1, \ldots, 30$$

Test Problem 4: Multimodality

$$f_1(x) = x_1$$
$$f_2(x) = g \times (1 - \sqrt{\frac{f_1}{g}})$$

$$g = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i))$$
$$x_1 \in [0,1], x_i \in [-5,5], i = 2,\ldots,10$$

Test Problem 5: Non–uniformity case

$$f_1(x) = 1 - \exp(-4x_2)\sin^6(6\pi x_1)$$
$$f_2(x) = g \times (1 - (\frac{f_1}{g})^2)$$
$$g(x) = 1 + 9(\frac{\sum_{i=2}^{n} x_i}{n-1})^{0.25}$$
$$x_i \in [0,1], i = 1,\ldots,10$$

All test problems contain two objective functions and thirty (first three problems) or ten (last two problems) 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 with a small mutation rate of 0.01. 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 maintain 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 this section, the solutions of five 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, PAESgray, PAES98gray, PAES98 and PAES98mut3p) using the statistical comparison technique.

In Figure 1(right) we plotted all non–dominated solutions for the first twenty runs of the five problems against the PAES98 and SPEA results. In addition, in Figure 1(left), the number of non–dominated solutions found for each crossover probability is shown. We obtained the algorithms' results from the web site

"http//www.tik.ee.ethz.ch/∽zitzler/testdata.html".

The results of PEAS was obtained from the web site

"http://www.rdg.ac.uk/∽ssr97jdk/multi/PAES.html"

The crossover rates of the solutions plotted were 0.15, 0.25, 0.05, 0.10, and 0.85 for the five test problems respectively. As can be seen in Figure 1 (left), the Pareto–frontier is always lower than SPEA and the distribution of the points on the Pareto–frontier is more uniformly distributed than SPEA. In comparison to PAES, the performance is similar with small improvement. However, for problem 4, PDE clearly outperformed PAES. The statistical test will give a more accurate conclusion later in this section.

In terms of the number of non–dominated solutions, one can see a trend in the first four problems, where the distribution of non–dominated solutions against the crossover rate follows a skewed bell shape. For problem 5, the peaks are close to each others and the bell shape is not clear. However, we will see in the rest of the discussions that the performance of PDE on problem 5 was somewhat consistent with small variations. SPEA had 179, 107, 190, 84, and 3 non–dominated solutions in total in all twenty runs.

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 most of the problems which may suggest that the relationship between the crossover rate and the solution's quality is almost unimodal. This is very interesting since it makes the search problem for finding a good crossover rate easy. However, extending our generalizations to any multiobjective problem would require many researchers to adopt PDE in a large number of real life applications.

To compare between PDE and the nine algorithms, we need to perform the statistical analysis using Knowles and Corne method [9]. Here, we use 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.

In Figures 2, 3, 4, 5, and 6, the performance of PDE for each crossover rate is compared against the nine algorithms. Since we found 4 different versions for PAES, we compared against the four version; therefore we have eight

algorithms and four versions of PAES to comprise a total of 12 comparisons. For the other algorithms, the results are the best published results; therefore, the crossover rate on the x–axis does not reflect the crossover rate used in each algorithm, but the crossover rate used to obtain the results for PDE.

In Figure 2, we can see that there is always a crossover rate where PDE outperforms all other algorithms. In problem 1, the best crossover rate is 0.15. This crossover rate achieves at least 80% dominance to all algorithms.

For problem 2, as shown in Figure 3, PDE outperformed all algorithms at crossover 0.25 except for SPEA where the statistical results are [17.7 20.1] which indicates that SPEA outperforms PDE with around 3% more.

For problem 3, as shown in Figure 4, although there is always a crossover rate where PDE is better than any of the twelve algorithms, there is no single crossover rate which is consistently better. We can verify this when comparing PDE against SPEA, where PDE outperforms SPEA with crossover rate 0.05 but PAES98 outperforms PDE at this crossover rate. However, at crossover 0.45 and 0.50, PDE outperforms PAES98 although it is very bad when compared against SPEA.

For problem 4, as shown in Figure 5, PDE outperformed all algorithms with any crossover rate less than 0.4. The best performance for PDE achieved with crossover rate of 0.10.

For problem 6, as shown in Figure 6, there is a clear trend that PDE dominated all other algorithms, except PAES98, for almost all crossover rate and dominated PAES98 at crossover rates of 0.80 and 0.85.

In Figure 7, we plotted the number of objective evaluations (minimum, average, and maximum) for each crossover probability for each problem. All problems except Problem 6 achieved small standard deviation in terms of the number of objective evaluations required at the best chosen crossover rate. In Problem 6, the best crossover rate was 0.85 and we can see here that there is some variations although since the average is close to the minimum than the maximum, this indicates that the maximum is somewhat an outlier. In brief, the best results were obtained with a total number of objective evaluations ranging between 35,000 to 50,000 for the five problems.

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 generated from a Gaussian distribution. We tested the approach on five benchmark problems and it was found that our approach is competitive to most other approaches. We also experimented with different crossover rates, on these five test problems, to find their best solutions. The crossover rates are found to be very sensitive to the solutions. However, in all cases, the crossover rate which results in a large number of non–dominated solutions also gives the best approximation to the pareto–front.

Acknowledgements

The authors would like to present their sincere gratitude to David Corne and Joshua Knowles for providing the MOSTAT code and making their results available on their webpage. Also, we would like to thank Eckart Zitzler for making the results of eight algorithms available on his webpage and Carlos Coello for maintaining the evolutionary multiobjective web site. Without the efforts of these prominent researchers, it would have been very difficult to compare our results. Also, we would like to thank Xin Yao and Bob McKay for discussing our ideas for evolving neural networks using VOPs.

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

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

[3] D.B. Fogel. *Evolutionary Computation: towards a new philosophy of machine intelligence.* IEEE Press, New York, NY, 1995.

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

[5] D.E. Goldberg. *Genetic algorithms: in search, optimisation and machine learning.* Addison Wesely, 1989.

[6] P. Hajela and C.Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.

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

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

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

[10] J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. *Genetic Algorithms and their Applications: Proceedings*

of the First International Conference on Genetic Algorithms, pages 93–100, 1985.

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

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

[13] E. Turban and J. Meredith. *Fundamentals of Management Science*. McGraw-Hill, Boston, USA, 1994.

[14] D. Van Veldhuizen and G. Mamont. Multiobjective evolutionary algorithm test suites. *Procedings of the 1999 ACM Sysposium on Applied Computing, San Antonio, Texas, ACM*, pages 351–357, 1999.

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

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

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

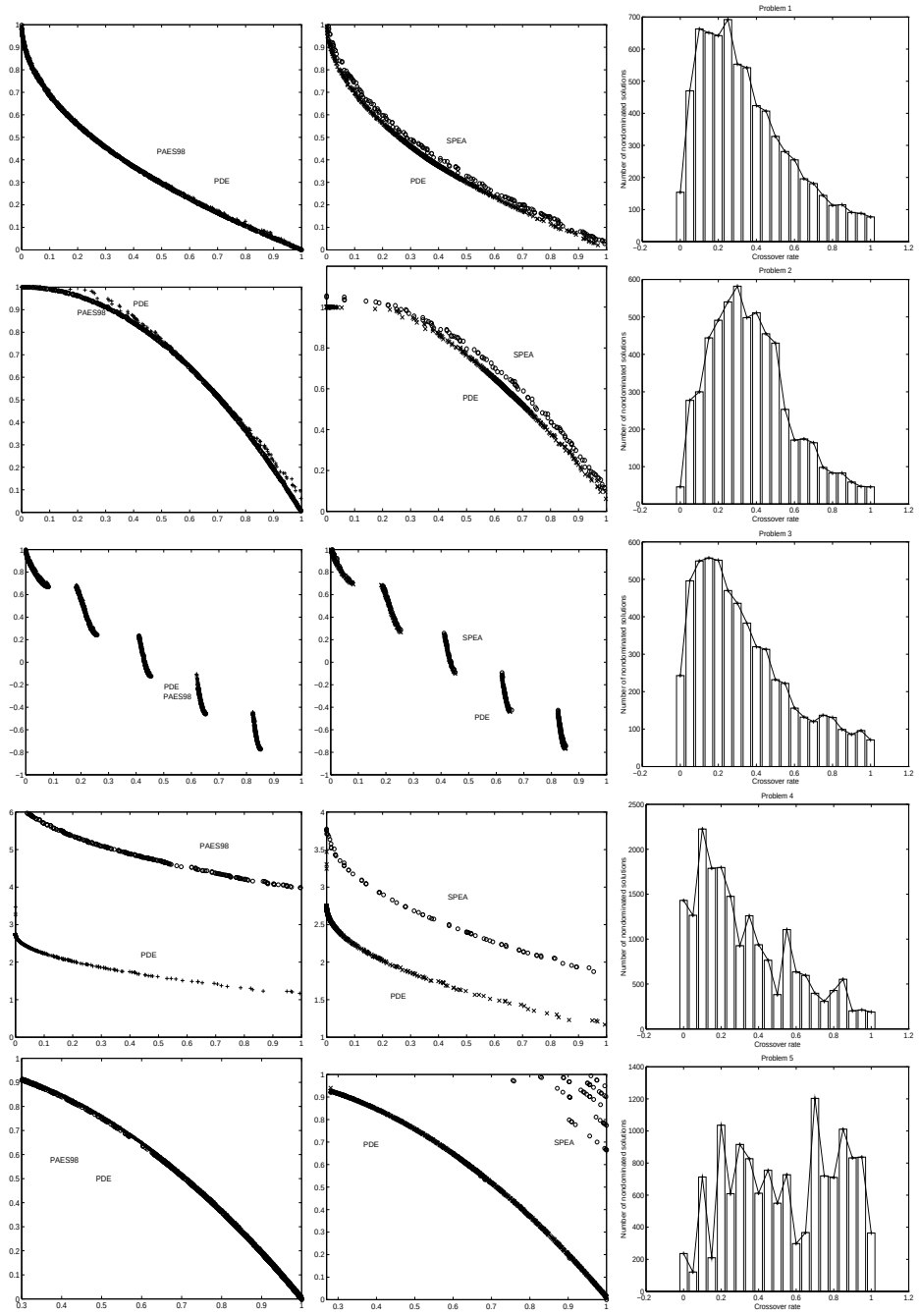

Figure 1: Right: The performance of the PDE algorithm compared with PAES98 and SPEA on the five test problems. Left: The distribution of the number of non–dominated solutions found by PDE in each problem.

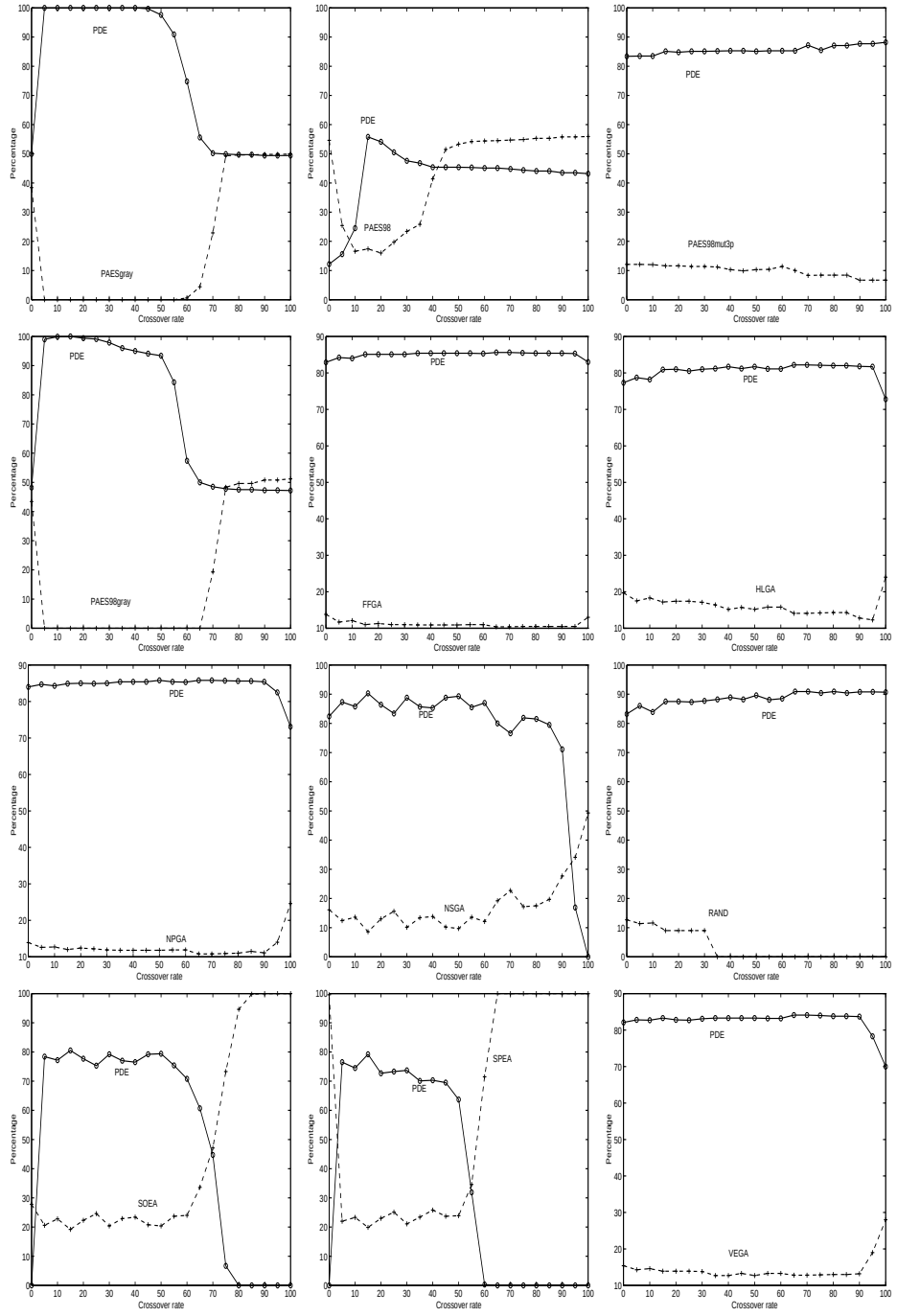

Figure 2: The percentage outperformed by PDE and the other algorithms for test problem 1. The x–axis represents the crossover rate for our algorithm and the y–axis represents the percentage outperformed by each algorithm.

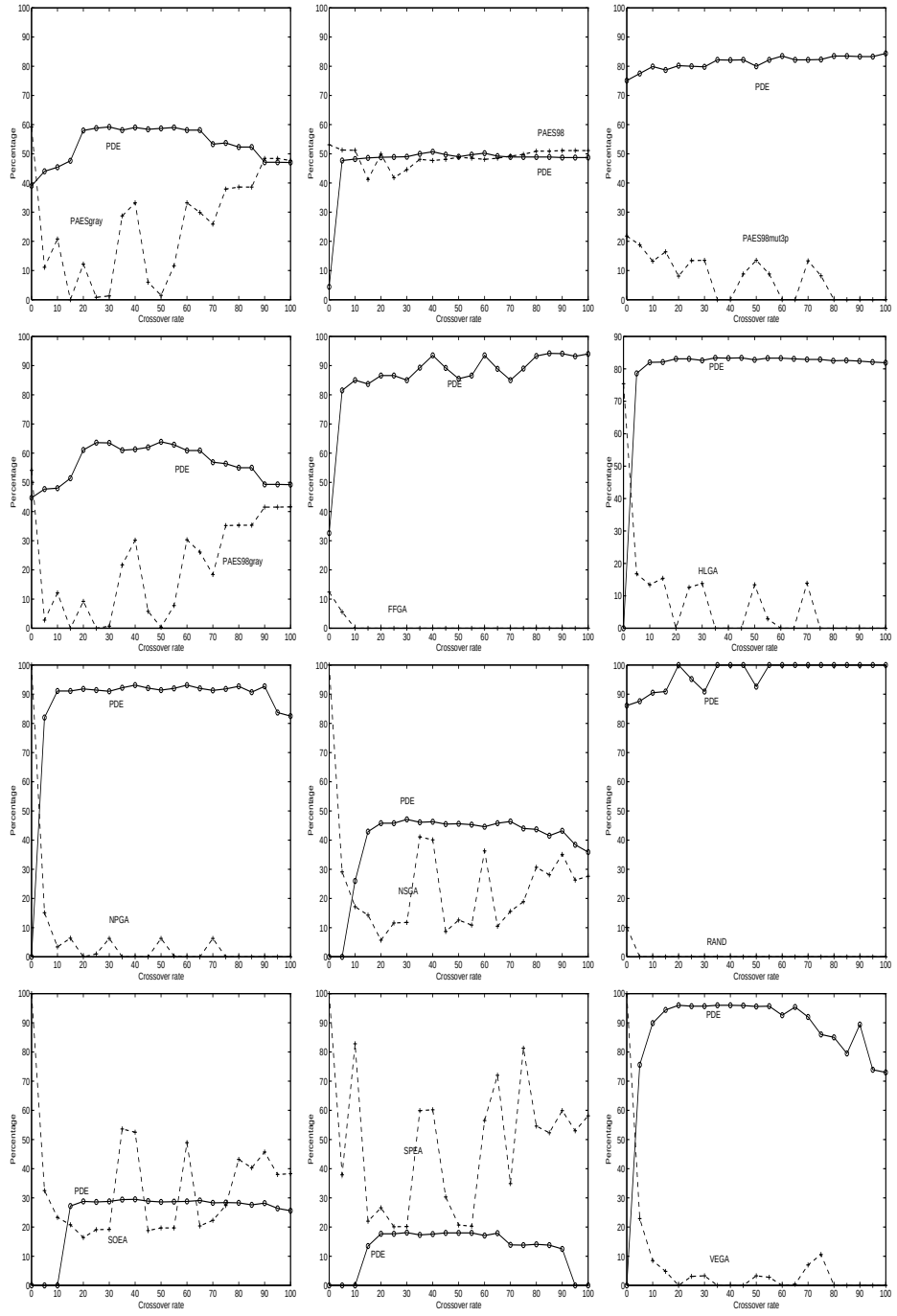

Figure 3: The percentage outperformed by PDE and the other algorithms for test problem 2. The x–axis represents the crossover rate for our algorithm and the y–axis represents the percentage outperformed by each algorithm.

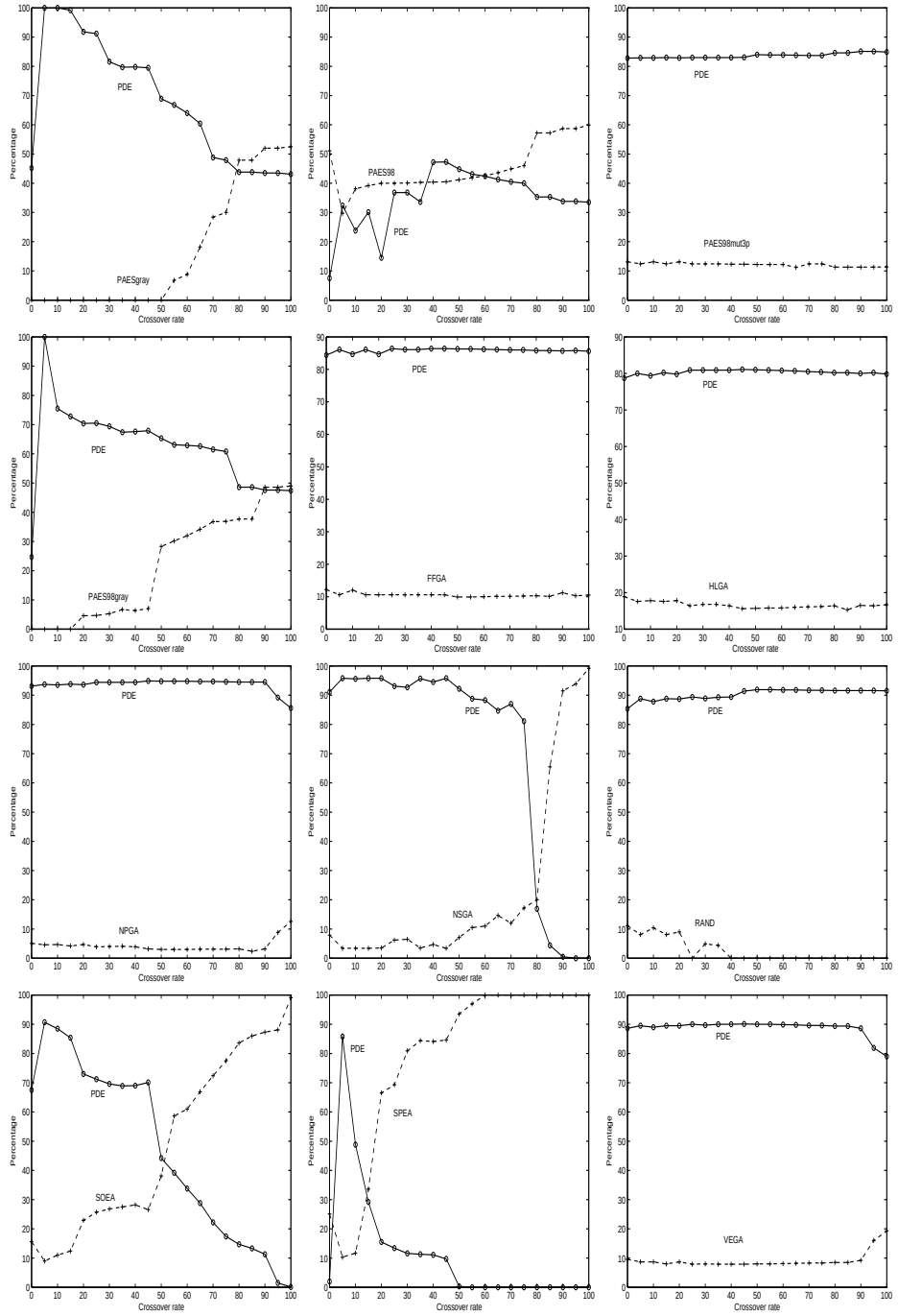

Figure 4: The percentage outperformed by PDE and the other algorithms for test problem 3. The x–axis represents the crossover rate for our algorithm and the y–axis represents the percentage outperformed by each algorithm.

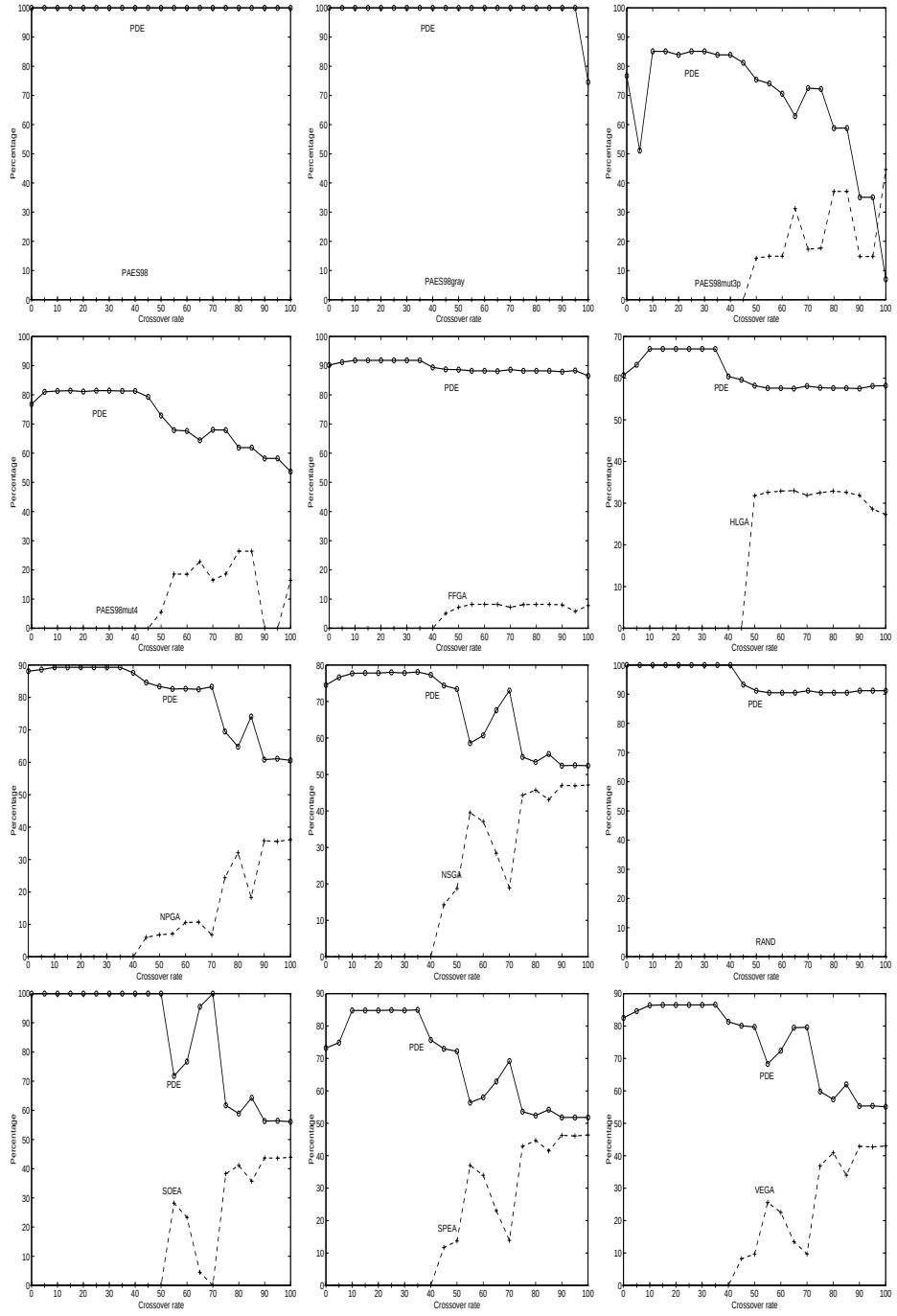

Figure 5: The percentage outperformed by PDE and the other algorithms for test problem 4. The x–axis represents the crossover rate for our algorithm and the y–axis represents the percentage outperformed by each algorithm.

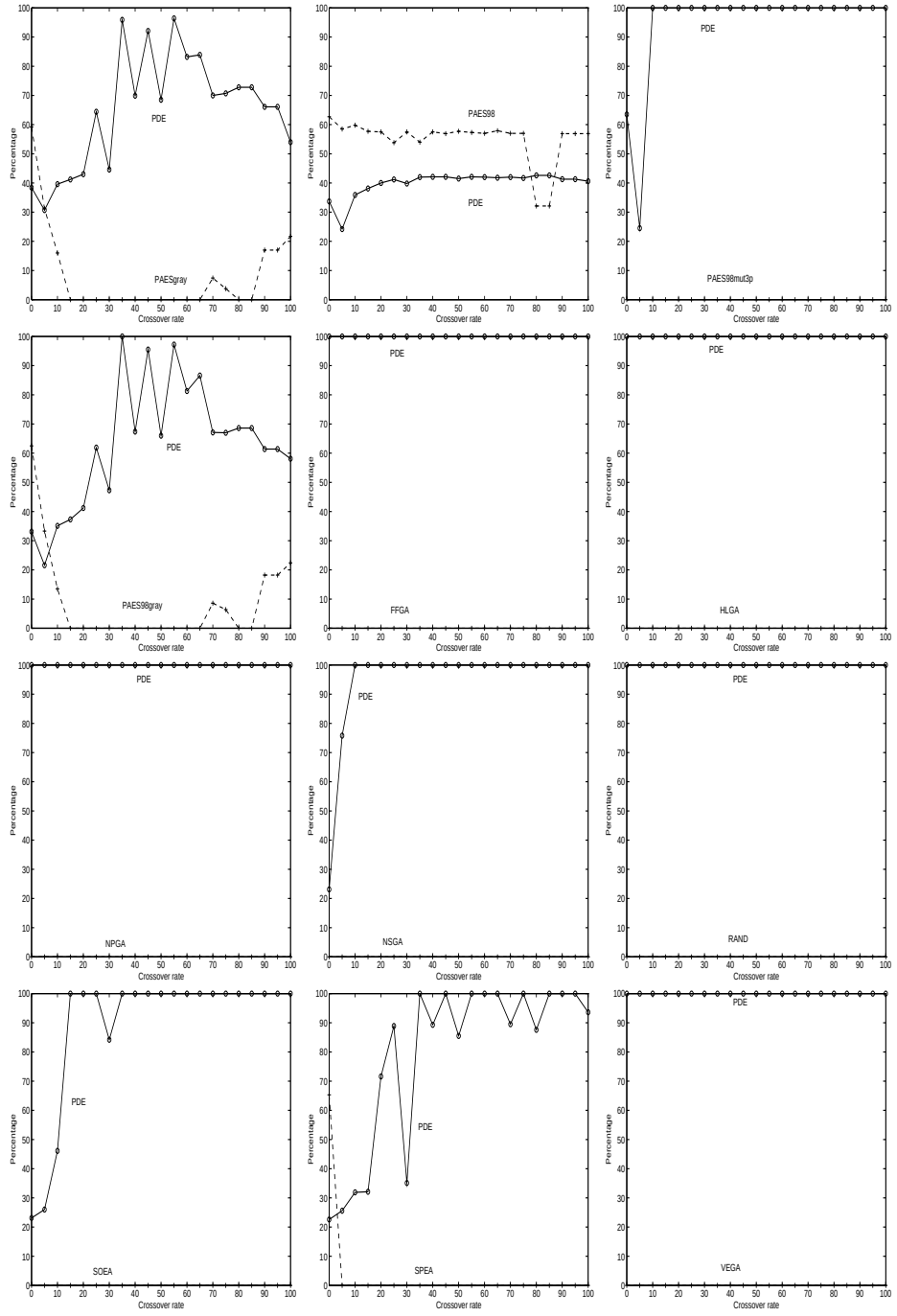

Figure 6: The percentage outperformed by PDE and the other algorithms for test problem 5. The x–axis represents the crossover rate for our algorithm and the y–axis represents the percentage outperformed by each algorithm.

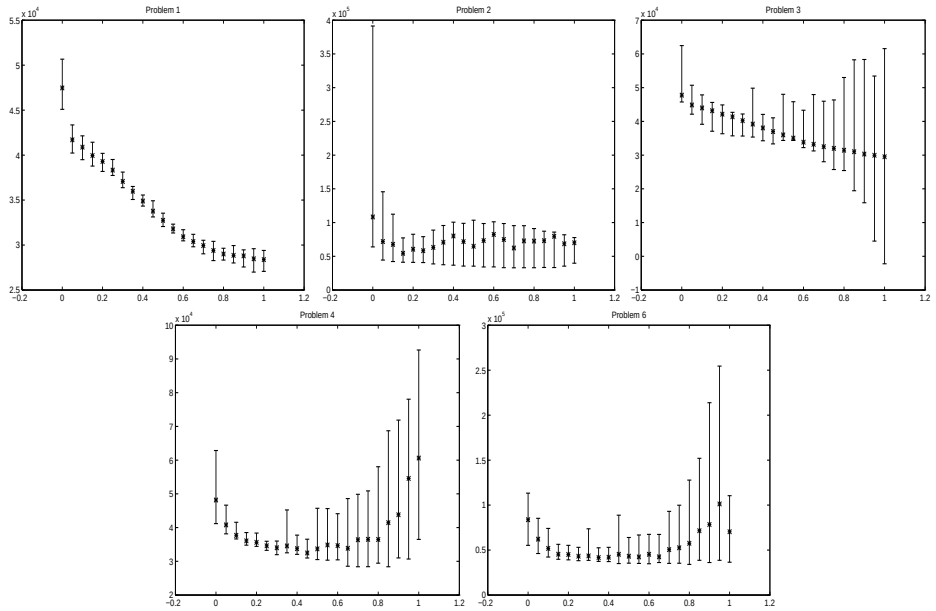

Figure 7: The minimum–average–maximum number of objective evaluations in each crossover operator for each problem.