Differential Evolution for Solving Multi-objective Optimization Problems

Ruhul Sarker and Hussein A. Abbass
School of Information Technology and Electrical Engineering,
University of New South Wales, ADFA Campus,
Northcott Drive, Canberra ACT, 2600, Australia,
{h.abbass,r.sarker,c.newton}@adfa.edu.au

Differential Evolution for Solving Multi-objective Optimization Problems

Abstract

The use of evolutionary strategies (ESs) to solve problems with multiple objectives (known as *Vector Optimization Problems* (VOPs)) has attracted much attention recently. Being population based approaches, ESs offer a means to find a set of Pareto-optimal solutions in a single run. *Differential Evolution* (DE) is an ES 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 VOPs. 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 VOPs.

1 Introduction

Although single objective decision models are sufficient for some decision making processes, there are many situations where decisions have multiple objectives. For example, we may need to maximize the profit while minimizing overtime. One may notice here that both objectives are conflicting in the normal case. Assuming that the normal working hours are utilized very efficiently, working overtime would increase the output and if the market needs the additional output, the profit is expected to increase.

Multi-objective problems are known as *Vector optimization problems* (VOPs). In these situations, the aim is to simultaneously optimize a set 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 [1].

Recently, *evolutionary strategies* (ESs) were found useful for solving VOPs [16]. ESs have some advantages over traditional OR techniques. For example, considerations for convexity, concavity, and/or continuity of functions are not necessary in ESs, whereas, they form a real concern in traditional OR techniques. Although ESs 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 develop a novel *Differential Evolution* (DE) algorithm for VOPs. The approach shows promising results when compared with the *Strength Pareto Evolutionary Algorithm* (SPEA) [16], for two benchmark problems. However there are several other known methods such as Fonseca and Fleming's genetic algorithm (FFGA)[5], Hajela's and Lin's genetic algorithm (HLGA) [7], Niched Pareto Genetic Algorithm (NPGA) [8], Non-dominated Sorting Genetic Algorithms (NSGA) [13], Random Sampling Algorithm (RAND) [16], Single Objective Evolutionary Algorithm (SOEA) [16], Vector Evaluated Genetic Algorithm (VEGA) [12] and Pareto Archived Evolution Strategy (PAES) [10] and [11]. There are several versions of PAES like PAES, PAES20, PAES98 and PAES98mut3p. We also 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 [10] and [11]. 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: 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 VOPs

Consider a VOP model as presented below:-

$$Optimize \quad F(\Omega) \tag{1}$$

$$\Omega = \{\vec{x} \in R^n | G(\vec{x}) \leq 0\} \tag{2}$$

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. These constraints can be lower and upper bounds on the variables.

In VOPs, the aim is to find the solution $\vec{x}^* \in \Omega$ which optimizes $F(\vec{x})$. 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, $\ncong$ and $\precnsim$ and then assume two vectors, $\vec{x}$ and $\vec{y}$. $\vec{x} \ncong \vec{y}$ 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 $\vec{x} \ncong \vec{y}$. $\ncong$ 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)

$$B_\delta(\vec{x}^*) = \{\vec{x} \in R^n |\ ||\vec{x} - \vec{x}^*|| < \delta\} \tag{3}$$

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". The following subsection provides background materials in evolutionary multi–objective.

2.2 VOPs and ESs

ESs for VOPs [1] 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 suffers from major drawbacks including: it produces a solution at a time, it assumes convexity of the pareto-frontier, and finding the right set of weights is not obvious. The simultaneous optimization can fit nicely with population based approaches, such as ESs, because they generate multiple solutions in a single run.

The Vector Evaluated Genetic Algorithm (VEGA) [12] is a population-based non-Pareto approach. In this approach, the total population is divided into a number of sub–populations equal to the number of objective functions to be optimized. Each sub–population is used to optimize each objective function independently. The sub–populations are then shuffled together followed by conventional crossover and mutation operators. Schaffer [12] realized that the solutions generated by his system were non-dominated with respect to the current sub–population, but they may not be the true non-dominated set.

In the Pareto-based approaches, the dominated and non-dominated solutions in the current population are separated. Goldberg [6] suggested a non-dominated ranking procedure to decide the fitness of the individuals. Later, Srinivas and Dev [13] 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. Recently, Deb and Goel ([4]) presented a revised version of NSGA, known as NSGAII, where the extent of elitism is controlled by fixing a user-defined parameter.

Fonseca and Fleming [5] 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. [8] proposed the *Niched Pareto Genetic Algorithm* (NPGA) that directly uses a set 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 only one of the two individuals is non–dominated by the individuals in the reference set, it is declared as the winner and is added to the mating pool. If both individuals are either dominated or non-dominated by the set, the

number of individuals in the mating pool that are within some pre–defined distance from each of the two individuals is calculated. The individual with the smallest number is added to the mating pool.

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, Zitzler and Thiele [16] 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 updates the 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.

SPEA has been recognized as a reference algorithm by many research in the last few years ([15], [2] and [9]). Recently, Zitzler et al. ([17] presented a revised version of SPEA which is known as SPEA2. The main differences of SPEA2 compared to SPEA are:

1. an improved fitness assignment scheme is used, which takes for each individual into account how many individuals it dominates and it is dominated by.
2. a nearest neighbor density estimation technique is incorporated which allows a more precise guidance of the search process.
3. a new archive truncation methods guarantees the preservation of boundary solutions.

As reported by Zitzler et al. ([17]), SPEA2 provides good performance in terms of convergence and diversity, and outperforms SPEA.

Most recently, Knowles and Corne [10, 11] 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 crowdedness 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 moves from a current solution to a nearby neighbor. As they reported, the algorithm works well, specially for problems of low computational complexity.

Knowles and Corne [11] 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. 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 intersecting 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 ($\alpha = 0.1$), and the percentage of the surface in which algorithm B significantly outperforms algorithm A. Before presenting our proposed algorithm, we need to present the Differential Evolution heuristic.

In the evolutionary multiobjective literature, there are a large number of methods and techniques. For comprehensive survey, the reader is advised to refer to [3].

2.3 Differential Evolution

DE is a branch of evolutionary algorithms developed by Storn and Price [14] 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. If there are boundary constraints, it is guaranteed, by some repair rules (Equation 6), 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 (Equation 5). 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 *Genetic Algorithms* (GAs) in a number of points:

1. DE uses real number representation while the 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 (Equation 5). 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 solution vectors ($M > 4$ since we need at least 3 parents for recombination and an additional one as a reference parent). The initial population, $P_{G=0} = \{\vec{x}^{1}_{G=0}, \ldots, \vec{x}^{M}_{G=0}\}$, is initialised as

$$x^{l}_{i,G=0} = lower(x_i) + rand_i[0,1] \times (upper(x_i) - lower(x_i)), \;\; l = 1, \ldots, M, \quad i = 1, 2, \ldots, N \tag{4}$$

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 initialised 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 $j \in [1, M]$. It is worth noting that DE is using an elitist approach, where parents are the elites in previous generations. The values of each variable in the child are changed with some crossover probability, CR, to

$$\forall i \leq N, x'_{i,G=k} = \begin{cases} x^{r_3}_{i,G=k-1} + F \times (x^{r_1}_{i,G=k-1} - x^{r_2}_{i,G=k-1}) & if \;\; (random[0,1) < CR \wedge i = i_{rand}) \\ x^{j}_{i,G=k-1} & otherwise \end{cases} \tag{5}$$

where $F \in (0, 1)$ is an algorithm parameter representing the amount of perturbation added to the main parent. The new solution replaces the old one if the new solution is better than the old one. During crossover, at least one of the variables should be changed to guarantee that the child is different from the parents if all crossover does not take place. This is represented in the algorithm by randomly selecting a

variable, $i_{rand} \in (1, N)$. This variable is forced to be crossed-over. After crossover, if one or more of the variables in the new solution are outside their boundaries, the following repair rule is applied

$$x'_{i,G=k} = \begin{cases} \frac{x^j_{i,G=k}+lower(x_i)}{2} & if \;\; x^j_{i,G=k+1} < lower(x_i) \\ lower(x_i) + \frac{x^j_{i,G=k}-upper(x_i)}{2} & if \;\; x^j_{i,G=k+1} > upper(x_i) \\ x^j_{i,G=k+1} & otherwise \end{cases} \tag{6}$$

The DE algorithm is presented in Figure 1.

---

let G denote 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 $P_{G=0} = \{\vec{x}^1_{G=0}, \ldots, \vec{x}^N_{G=0}\}$ *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 (eg fixed number of generations) 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 $i \leq N, x'_{i,G=k} =$
$\begin{cases} x^{r_3}_{i,G=k-1} + F \times (x^{r_1}_{i,G=k-1} - x^{r_2}_{i,G=k-1}) & if \;\; (random[0,1) < CR \wedge i = i_{rand}) \\ x^j_{i,G=k-1} & otherwise \end{cases}$
end forall
$\vec{x}^j_{G=k} = \left\{ \begin{array}{ll} \vec{x}'_{G=k} & if\ f(\vec{x}'_{G=k}) \leq f(\vec{x}^j_{G=k-1}) \\ \vec{x}^j_{G=k-1} & otherwise \end{array} \right\}$
end forall
$k = k + 1$
evaluate $P_{G=i}$
end while
return *the best encountered solution* x.

Figure 1: The Differential Evolution Algorithm

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

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:-

let G denote 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 $P_{G=0} = \{\vec{x}^1_{G=0}, \ldots, \vec{x}^N_{G=0}\}$ *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 (eg fixed number of generations) 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 $i \leq N, x'_{i,G=k} =$

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

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**

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

$j = j + 1$

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)

1. The initial population is initialized according to a Gaussian distribution $N(0.5, 0.15)$. The variables are generated within their boundaries [0,1]; otherwise the repair rule is used.

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. Assuming that the variables are bounded between $[a, b]$. If a is positive and the variable is a negative value, the sign of the variable is reversed. If the variable is less than a, a value of 1 is added until it is greater than of equal to a. If the variable is greater than b, a value of 1 is subtracted until it is less than b. The assumption here is that $b - a$ is greater than 1.

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

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:

$$D(x) = \frac{(\min_{\forall i \in N} ||x - x_i|| + min_{\forall j \in N, j \neq i} ||x - x_j||)}{2}, \tag{7}$$

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 Zitzler and Thiele [16]:

Test Problem 1: Convex

$$f_1(x) = x_1 \tag{8}$$

$$f_2(x) = g \times (1 - \sqrt{(\frac{f_1}{g})}) \tag{9}$$

$$g = 1 + 9 \times \frac{(\sum_{i=2}^{n} x_i)}{(n-1)} \tag{10}$$

$$x_i \in [0, 1], i = 1, \ldots, 30 \tag{11}$$

Test Problem 2: Discontinuous Pareto–frontier

$$f_1(x) = x_1 \tag{12}$$

$$f_2(x) = g * (1 - \sqrt{\frac{f_1}{g}} - (\frac{f_1}{g}) \sin(10\pi f_1)) \tag{13}$$

$$g = 1 + 9 \times \frac{(\sum_{i=2}^{n} x_i)}{(n-1)} \tag{14}$$

$$x_i \in [0, 1], i = 1, \ldots, 30 \tag{15}$$

Both test problems contain two objective functions and thirty variables. The first is a conventional VOP with convex and continuous Pareto frontier; although the number of Pareto optimal solutions is extremely large. The second problem cannot be handled with the traditional approaches (eg weighted sum method) because of the number of Pareto solutions and the discontinuity of the Pareto frontier. 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 within the variable's 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. We used the same experimental setup of the other algorithms. Therefore, the total relative time (as estimated with the number of objective evaluations) is almost the same.

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".

In Figure 3, we plotted all the non-dominated solutions for the first twenty runs of both test problems with the best SPEA results obtained from the web site. The crossover rates of the solutions plotted were 0.15 and 0.05 for the first and second test problems respectively for our algorithm. SPEA results are the best published ones. 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.

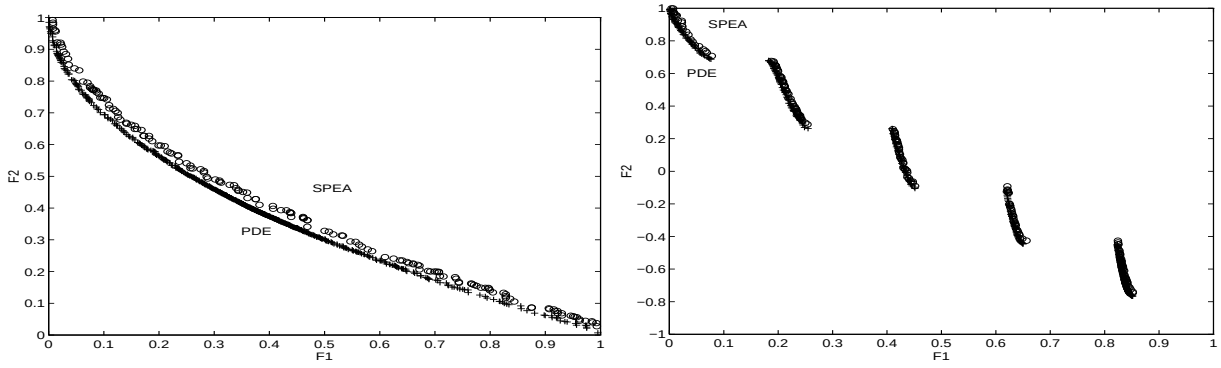

Figure 3: The performance of the PDE algorithm compared with SPEA on the test problem.

To perform the statistical analysis using the Knowles and Corne method [11], we used the solutions of the twenty runs for each crossover rate. The results of the comparison are 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.

The percentage outperformed by our algorithm and the twelve other algorithms are plotted against the crossover rate in Figure 4 and 5 for both test problems. For the other twelve algorithms, the results are the best published results; therefore, the crossover rate on the x-axis does not reflect the crossover rate used in them. 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 6. 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.

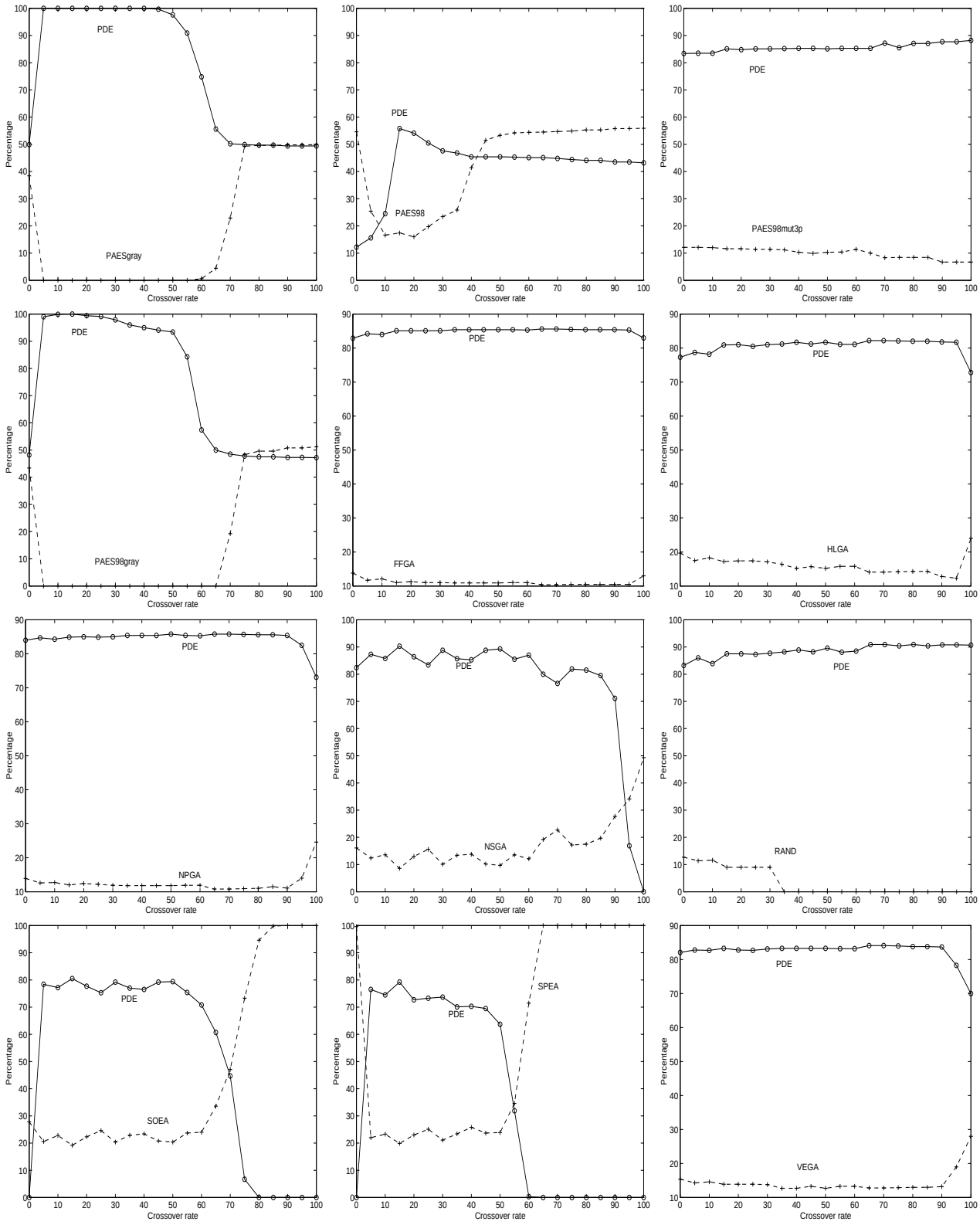

Figure 4: The percentage outperformed by PDE and the other algorithms for test problem 1. The x–axis represents the crossover rate for our algorithm and the y–axis represents the percentage outperformed by each algorithm.

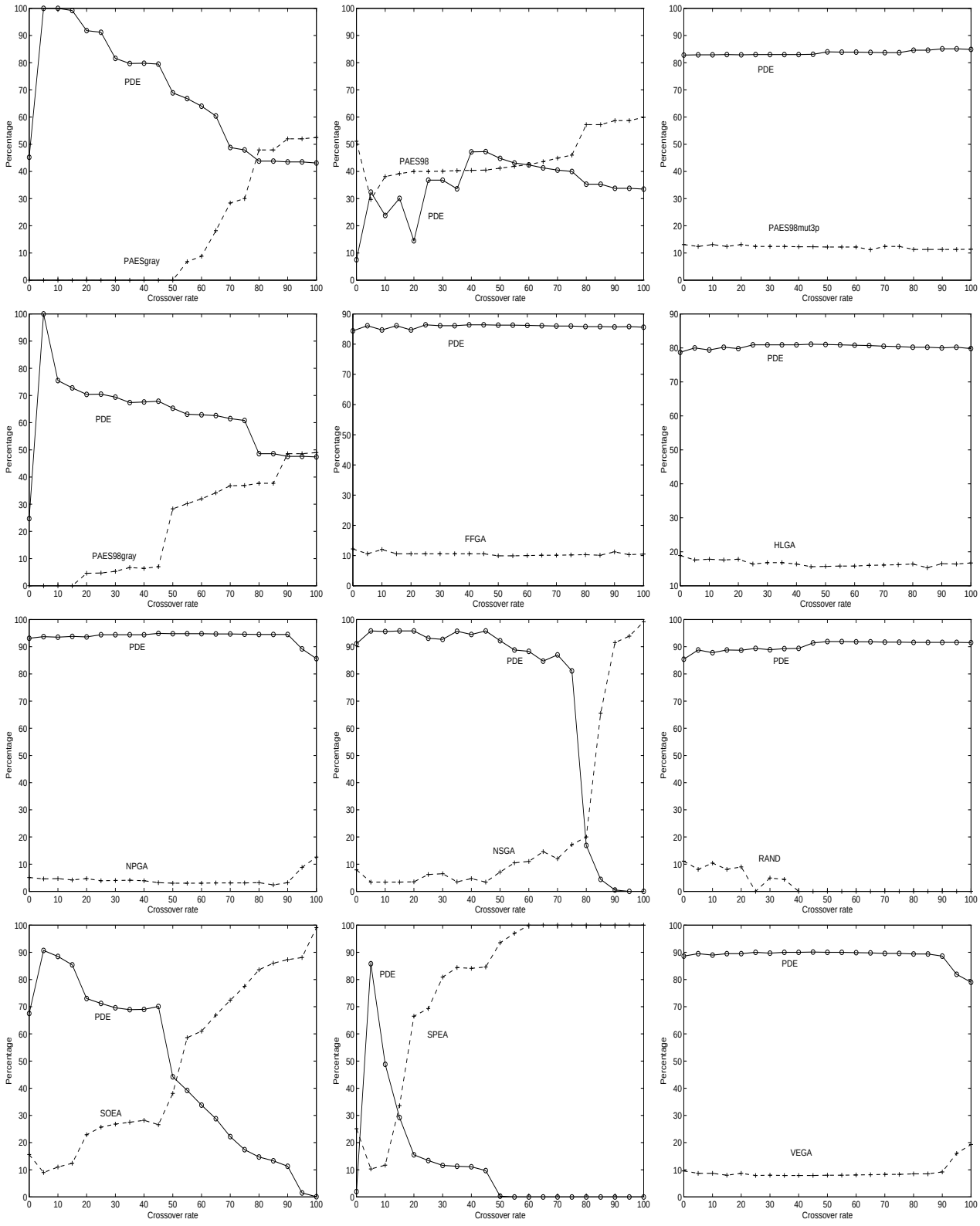

Figure 5: The percentage outperformed by PDE and the other algorithms for test problem 2. The x–axis represents the crossover rate for our algorithm and the y–axis represents the percentage outperformed by each algorithm.

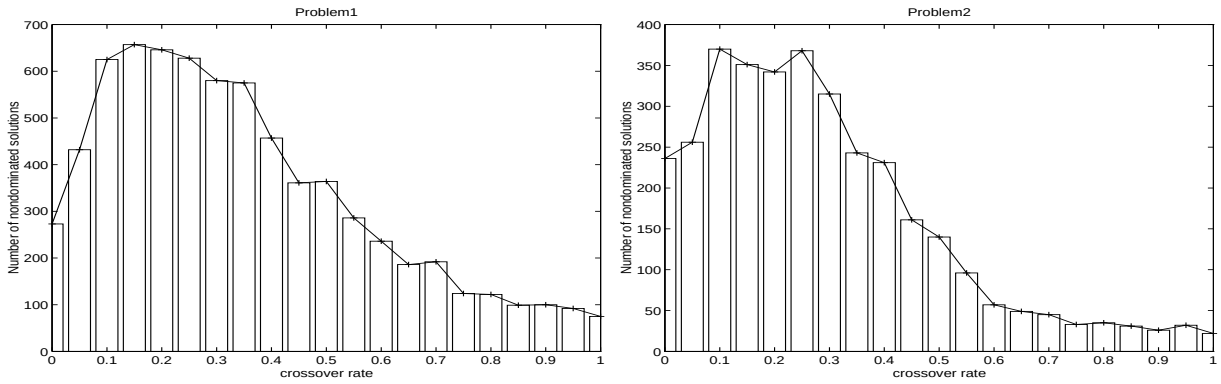

Figure 6: The distribution of the number of non-dominated solutions found by our algorithm for the two test problems using different crossover rates. The x-axis represents the crossover rate and the y-axis represents the number of non-dominated solutions found.

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 solution 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.

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 since 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. A trend was found which suggests that large number of non-dominated solutions were found with low-crossover rates. From our point of view, these are good news. Crossover in DE is somewhat a directed mutation operator as it mutates one parent by the difference of the other two. Hence, a good performance with low crossover rates entails that large steps was not useful in these problems.

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. Also, we anticipate applying the algorithm on real life decision making problems. This, however, will require the development of good user interfaces so that managers can interact with the package easily.

References

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

[2] D. Corne, J. Knowles and M. Oates. The pareto envelope-based selection algorithm for multiobjective optimization. *Parallel Problem Solving from Nature - PPSN VI*, Berlin, Springer, pages 839–848, 2000.

[3] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons,, New York, 2001.

[4] K. Deb and T. Goel. Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence. KanGAL Report No. 2000004, IIT Kanpur, India, 2002.

[5] 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.

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

[7] P. Hajela and C. Lin. Genetic Search Strategies in Multi-Criterion Optimal Design. *Structural Optimization*, 4:99–107, 1998.

[8] 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.

[9] A. Jaszkiewicz. On the performance of multiple objective genetic local search on the 0/1 knapsack problem: a comparative experiment. Technical Report RA-002/2000, Institue of Computing Science, Pozan University of Technology, 2000.

[10] 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.

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

[12] 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.

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

[14] 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.

[15] K. C. Tan, T. H. Lee and E. F. Khor. Incrementing multi-objective evolutionary algorithms: Performance studies and comparisons. In E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne (Eds.), Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO2001), Volume 1993 of Lecture Notes in Computer Science, Berlin, Springer-Verlag, pages 111–125, 2001

[16] 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.

[17] E. Zitzler, M. Laumanns and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. TIK-Report 103, ETH Zurich, Switzerland, 2001.

Authors' Biography

Ruhul Sarker received his Ph.D. in 1991 from DalTech, Dalhousie University, Halifax, Canada, and is currently a Senior Lecturer in Operations Research at the School of Information Technology and Electrical Engineering, University of New South Wales, ADFA Campus, Canberra, Australia. Before joining at UNSW in February 1998, Dr Sarker worked with Monash University, Victoria, and the Bangladesh University of Engineering and Technology, Dhaka. His main research interests are Evolutionary Optimization, Data Mining and Applied Operations Research. He has recently edited four books and has published more than 80 refereed papers in international journals (journal includes ITOR, EJOR, C&IE, IJIE, IJPE, JORS, AMM, APJOR, IJSS and others) and conference proceedings. He is the editor of ASOR Bulletin, the national publication of the Australian Society for Operations Research. Dr Sarker is actively involved with a number of national and international conferences & workshop organizations in the capacity of chair / co-chair or program committee member. He was also involved with a number of proceedings editorships. He has recently been selected as a general vice-chair and technical co-chair for IEEE World Congress on Evolutionary Computation 2003. He is also a member of task force for promoting evolutionary multiobjective optimization operated by IEEE Neural Network Society. For further details, please visit his website: http://www.itee.adfa.edu.au/ ruhul/

Hussein A. Abbass is a Senior Lecturer at the School of Information Technology and Electrical Engineering, University of New South Wales, ADFA Campus, Canberra, Australia. He is also an honorary associate with the University of New England, Australia. Dr. Abbass holds a PhD (QUT, Australia), a M.Sc. (University of Edinburgh, UK), a Research Masters in OR, PG-Diploma in OR, B.A. and B.Sc. (Cairo University, Egypt). Dr. Abbass has more than 14 years experience in industry and academia. His current teaching responsibilities include Data Mining, Knowledge Discovery in Databases, Modern Heuristic Techniques, and Computational Optimization. His research interest includes Neural Networks, Rule Discovery, Evolutionary single/multiobjective constrained/unconstrained Optimization, and Artificial Life and Robotics. Dr. Abbass is the chair of Task Force on Artificial Life and Complex Adaptive

Systems by IEEE Neural Network Society, is the vice-chair of the 2003 Congress on Evolutionary Computation, Co-chair of the 2003 Australian Conference on Artificial Life, Organising chair of the 2002 Australian Joint Conference on Artificial Intelligence, Organizing and program committee member of ALife 8, data mining stream chair of IFORS'02, and member of the program committee of many other conferences including GECCO'03, CEC'01&02, CIRAS'03, and others. He served as a guest editor for a number of books and journals including JASS, IJCIA, and Artificial Life. http://www.itee.adfa.edu.au/ abbass/