The Self–Adaptive Pareto Differential Evolution Algorithm

Hussein A. Abbass
School of Computer Science, University of New South Wales, ADFA campus, Canberra ACT, 2600, Australia, h.abbass@adfa.edu.au

Abstract -
The *Pareto Differential Evolution* (PDE) algorithm was introduced last year and showed competitive results. The behavior of PDE, as in many other *evolutionary multiobjective optimization* (EMO) methods, varies according to the crossover and mutation rates. In this paper, we present a new version of PDE with self-adaptive crossover and mutation. We call the new version Self–adaptive Pareto Differential Evolution (SPDE). The emphasis of this paper is to analyze the dynamics and behavior of SPDE. The experiments will also show that the algorithm is very competitive to other EMO algorithms.

I. Introduction

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 [11] for optimization problems over continuous domains.

The aim of *multiobjective optimization problems* (MOPs) is to generate a list (called the *pareto or non–dominated list*) of solutions for problems with more than one objective. Each solution A in the list is optimal in the sense that no other solution B of the problem better than A as measured by **all** the objectives.

Different *evolutionary multiobjective optimization* (EMO) methods have been proposed in the literature to overcome the drawbacks of traditional approaches to MOPs. EMO methods do not have assumptions underlying the MOP. In addition, most of them are population based; therefore they can generate a number of pareto solutions in a single run. One of the recent approaches to EMO is the *Pareto Differential Evolution* (PDE) algorithm [1]. The algorithm was designed for EMO problems with continuous variables and achieved a very competitive results compared to other algorithms in the EMO literature. However, there was no obvious way to select the best crossover and mutation rates apart from running the algorithm with different rates, then selecting the best among them. This is actually a problem with most EMO methods.

In this paper, we introduce a new version of PDE where the crossover and mutation rates are self-adaptive. We call the new version *Self–adaptive Pareto Differential Evolution* (SPDE). This paper aims to introduce SPDE, an initial analysis to its behavior and some comparisons against well known algorithms for EMO problems. The paper is organized as follows: background materials are covered in Section II followed by the proposed algorithm in Section III. Experiments are then presented in Section IV and conclusions are drawn in Section V.

II. Background Materials

Consider a MOP 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 MOPs, the aim is to find the optimal 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 MOPs, we need to define two operators, $\not\geq$ and $\precnsim$ and then assume two vectors, $\vec{x}$ and $\vec{y}$. $\vec{x} \not\geq \vec{y}$ iff $\exists\ x_i \in \vec{x}$ and $y_i \in \vec{y}$ such that $x_i \neq y_i$. And, $\vec{x} \precnsim \vec{y}$ iff $\forall\ x_i \in \vec{x}$ and $y_i \in \vec{y}, x_i \leq y_i$, and $\vec{x} \not\geq \vec{y}$. $\not\geq$ and $\precnsim$ can be seen as the "not equal to" and "less than or equal to" operators respectively, over two vectors. We can now define the concepts of pareto solutions in VOPs.

Efficient (non–inferior/ pareto–optimal) solution A vector $\vec{x}^* \in \Omega$ is said to be an efficient solution of

VOP iff $\nexists\ \vec{x}\ \in\ \Omega$ such that $F(\vec{x}) \precsim F(\vec{x}^*)$.

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

EMO methods usually fall into one of three categories, viz plain aggregating, non–Pareto and Pareto–based approaches. For an interested reader, a comprehensive survey is presented in [2]. Some of these methods include: *Random Sampling Evolutionary Algorithm* (RAND)[12], *Hajela's and Lin's genetic algorithm* (HLGA) [5], *single objective evolutionary algorithm* (SOEA) [12], *Vector Evaluated Genetic Algorithm* (VEGA) [9], *Non–dominated Sorting Genetic Algorithms* (NSGA) [10], *Fonseca and Fleming's genetic algorithm* (FFGA) [4], *Niched Pareto Genetic Algorithm* (NPGA) [6], and *Pareto Archived Evolution Strategy* (PAES) [7], [8].

To compare between different algorithms, we use Knowles and Corne [8] statistical analysis method. For a complete description of this method, the reader can refer to [8]. When comparing two algorithms A and B, the method outputs two values $[a, b]$. The value a represents the percentage of the pareto–frontier where algorithm A outperformed algorithm B and the value b represents the percentage of the pareto–frontier algorithm B outperformed algorithm A. The sum of a and b should not exceed 100. The value $100 - (a + b)$ represents the percentage of the pareto–frontier where both A and B are statistically insignificant at confidence level 0.95.

III. Self–adaptive Pareto Differential Evolution

The SPDE algorithm for vector optimization problems is an adaptation of the PDE algorithm described in [1]. Similar to PDE, the SPDE 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.

In contrast to PDE, SPDE self–adapts the crossover and mutation rates. Both rates are inherited from the parents in the same way crossover is undertaken for the decision variables. 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 (Equation 6), 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. **Crossover rate:** Let the crossover rate be

$$x_c^{child} \leftarrow x_c^{\alpha_1} + N(0,1) \times (x_c^{\alpha_2} - x_c^{\alpha_3}) \quad (1)$$

      If the crossover rate is not in $[0, 1]$, repair the crossover rate according to the repair rule.
      iv. **Mutation rate:** Let the mutation rate be

$$x_m^{child} \leftarrow x_m^{\alpha_1} + N(0,1) \times (x_m^{\alpha_2} - x_m^{\alpha_3}) \quad (2)$$

      If the mutation rate is not in $[0, 1]$, repair the mutation rate according to the repair rule.
      v. **Crossover:** For each variable i
      With some probability $Uniform(0, 1) > x_c^{child}$ or if $i = j$, do

$$x_i^{child} \leftarrow x_i^{\alpha_1} + N(0,1) \times (x_i^{\alpha_2} - x_i^{\alpha_3}) \quad (3)$$

      otherwise

$$x_i^{child} \leftarrow x_i^{\alpha_1} \quad (4)$$

      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.

vi. If the child dominates the main parent,

A. **Mutation:** With some probability $Uniform(0,1) > x_m^{child}$, do
For each variable i

$$x_i^{child} \leftarrow x_i^{child} + N(0, 0.1) \times range \quad (5)$$

Where *range* is the difference between the maximum value the variable can take and its minimum.

B. place the child into the population.

(f) Until the population size is M

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

If the maximum number of non–dominated solutions in the generation is greater than the user specified maximum, the following nearest neighbor distance function is adopted:

$$D(x) = \frac{(min||x - x^i|| + min||x - x^j||)}{2}, \quad (6)$$

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 the user specified maximum. 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.

IV. Experiments

A. Test Problems

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

Test Problem 1 (P1): Convex

$$f_1(x) = x_1 \quad (7)$$

$$f_2(x) = g \times (1 - \sqrt{\frac{f_1}{g}}) \quad (8)$$

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

$$x_i \in [0,1], i = 1, \ldots, 30 \quad (10)$$

Test Problem 2 (P2): Non–convex counterpart to test problem 1

$$f_1(x) = x_1 \quad (11)$$

$$f_2(x) = g \times (1 - (\frac{f_1}{g}))^2 \quad (12)$$

$$g = 1 + 9 \times \frac{\sum_{i=2}^{n} x_i}{n-1} \quad (13)$$

$$x_i \in [0,1], i = 1, \ldots, 30 \quad (14)$$

Test Problem 3 (P3): Multimodality

$$f_1(x) = x_1 \quad (15)$$

$$f_2(x) = g \times (1 - \sqrt{\frac{f_1}{g}}) \quad (16)$$

$$g = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i)) \quad (17)$$

$$x_1 \in [0,1], x_i \in [-5,5], i = 2, \ldots, 10 \quad (18)$$

Test Problem 4 (P4): Non–uniformity case

$$f_1(x) = 1 - \exp(-4x_2)\sin^6(6\pi x_1) \quad (19)$$

$$f_2(x) = g \times (1 - (\frac{f_1}{g})^2) \quad (20)$$

$$g(x) = 1 + 9(\frac{\sum_{i=2}^{n} x_i}{n-1})^{0.25} \quad (21)$$

$$x_i \in [0,1], i = 1, \ldots, 10 \quad (22)$$

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

B. Experimental Setup

To be consistent with the literature [12], the initial population size is set to 100 and the maximum number of objective evaluations is set to 25000. The initial population is initialized according to a Gaussian distribution $N(0.5, 0.15)$, assuming that the variables are bounded between [0,1]; otherwise the normal distribution is scaled to cover the variables' range. Therefore, with high probability, the Gaussian distribution will generate values between $0.5 \pm 3 \times 0.15$ which fits within the variables' boundaries. If a variable's value is outside its range, the repair rule is used to maintain the boundary constraints. Crossover and mutation rates are initialized in the initial population from a uniform distribution between $[0,1]$. 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. Each problem was repeated 20 times with 20 different seeds.

C. Experimental Results and Discussions

In this section, the solutions of four test problems using the proposed SPDE algorithm are compared with the solutions of thirteen other EMO algorithms (FFGA, NSGA, SPEA, HLGA, RAND, VEGA, NPGA, SOEA, PAES98, PAES.gray, PAES98.gray, PAES20 and PAES98mut3p) using the statistical comparison technique of Knowles and Corne [8]. 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". Tables I and II presents the SPDE results without and with mutation respectively. By comparing both tables, the results without mutation as shown in Table I are much better than their counterparts in Table II for problems P1, P2, and P4. For problem P3, we can see a slight improvement for SPDE with mutation against all algorithms except for VEGA and SOEA. We can also notice that there is a considerable improvement against PAES.gray.

TABLE I

THE RESULTS OF THE STATISTICAL ANALYSIS PERFORMED BETWEEN SPDE AND 13 OTHER ALGORITHMS. THE VALUES TAKE THE FORM $[a, b]$, WHERE a IS THE PERCENTAGE OF THE PARETO FRONTIER WHERE SPDE OUTPERFORMED THE OTHER ALGORITHM AND b IS THE PERCENTAGE THE OTHER ALGORITHM OUTPERFORMED SPDE FOR THE FOUR TEST PROBLEMS. NO MUTATION IS USED WITH SPDE.

Algorithm	Functions			
	P1	P2	P3	P4
FFGA	[85.1,11.0]	[81.5,10.1]	[85.6,9.8]	[100.0,0.0]
NSGA	[85.9,13.2]	[83.0,15.8]	[74.7,20.9]	[100.0,0.0]
SPEA	[70.7,23.5]	[76.4,23.0]	[68.5,13.9]	[100.0,0.0]
HLGA	[81.1,17.2]	[81.4,16.4]	[55.7,41.5]	[100.0,0.0]
RAND	[87.6,7.5]	[87.7,7.4]	[83.8,12.1]	[100.0,0.0]
VEGA	[83.3,13.3]	[91.9,7.3]	[80.1,14.4]	[100.0,0.0]
NPGA	[84.9,12.0]	[90.4,7.8]	[81.7,15.4]	[100.0,0.0]
SOEA	[77.0,22.1]	[77.9,22.0]	[100.0,0.0]	[100.0,0.0]
PAES98	[45.5,43.0]	[57.6,0.0]	[70.7,0.0]	[98.6,0.7]
PAES.gray	[100.0,0.0]	[100.0,0.0]	[68.9,0.0]	[100.0,0.0]
PAES98.gray	[96.5,0.0]	[100.0,0.0]	[100.0,0.0]	[100.0,0.0]
PAES20	[48.1,18.0]	[51.9,0.0]	[72.0,0.0]	[99.4,0.0]
PAES98.mut3p	[85.1,11.4]	[77.2,20.0]	[67.5,25.0]	[100.0,0.0]

C.1 Analyzing the Dynamics of SPDE

In this section, we are interested in analyzing the dynamics and behavior of SPDE. To undertake this, we plotted in Figure 1 the first eight generations of a typical run for SPDE. By scrutinizing the top two graphs in the figure, it is easy to see a dramatic improvement from generation 0 to generation 1. In generation 0, the solutions were generated at random and all pareto optimal solutions in the population were very far from the actual pareto frontier. A substantial shift of the pareto frontier occurred between generation 0 to generation 1. In addition, the number of pareto optimal solutions is very small (around 15 pareto solutions) compared to the dramatic increase in generation 1. The intensity of the pareto frontier in generation 1 is supported by the thickness of the curve. It is notable however that the pareto frontier became sparse once more in generations 3 to 6 where it started thickening again in generation 7. On the contrary to the transition from generation 0 to 1, starting from generation 1, the length of the shift in the pareto frontier in each generation towards the actual pareto frontier is quite small.

The dynamics of SPDE entail that the algorithm flips all the time between improving the quality of the pareto frontier by pushing it towards the actual one, and improving the intensity of the number of pareto optimal solutions on the pareto frontier. This type of dynamics is quite interesting as it suggests the possibility of developing a two phase algorithm where the first stage pushes the pareto set to the actual pareto frontier and the second stage spread the solutions and increase the intensity of the pareto solutions on the pareto frontier.

TABLE II

THE RESULTS OF THE STATISTICAL ANALYSIS PERFORMED BETWEEN SPDE AND 13 OTHER ALGORITHMS. MUTATION IS USED WITH SPDE.

Algorithm	Functions			
	P1	P2	P3	P4
FFGA	[84.7,11.3]	[81.0,10.4]	[85.6,8.7]	[100.0,0.0]
NSGA	[64.7,28.3]	[69.7,29.3]	[74.7,20.4]	[100.0,0.0]
SPEA	[0.0,100.0]	[54.8,41.7]	[68.4,13.9]	[100.0,0.0]
HLGA	[80.2,17.8]	[80.5,17.8]	[56.1,39.4]	[100.0,0.0]
RAND	[87.4,7.6]	[87.5,7.5]	[86.8,7.5]	[100.0,0.0]
VEGA	[81.9,13.9]	[76.7,22.7]	[80.0,14.4]	[100.0,0.0]
NPGA	[84.3,12.5]	[83.6,14.7]	[82.4,12.1]	[100.0,0.0]
SOEA	[0.0,100.0]	[61.6,35.8]	[100.0,0.0]	[100.0,0.0]
PAES98	[43.4,55.7]	[41 1,3.9]	[71.0,0.0]	[98.3,1.2]
PAES.gray	[49.5,49.7]	[86.9,3.3]	[100.0,0.0]	[100.0,0.0]
PAES98.gray	[47.4,50.8]	[89.1,2.4]	[100.0,0.0]	[100.0,0.0]
PAES20	[44.8,54.6]	[36.7,14.9]	[75.4,0.0]	[98.9,0.0]
PAES98.mut3p	[84.9,11.6]	[76.8,20.2]	[68.0,23.8]	[100.0,0.0]

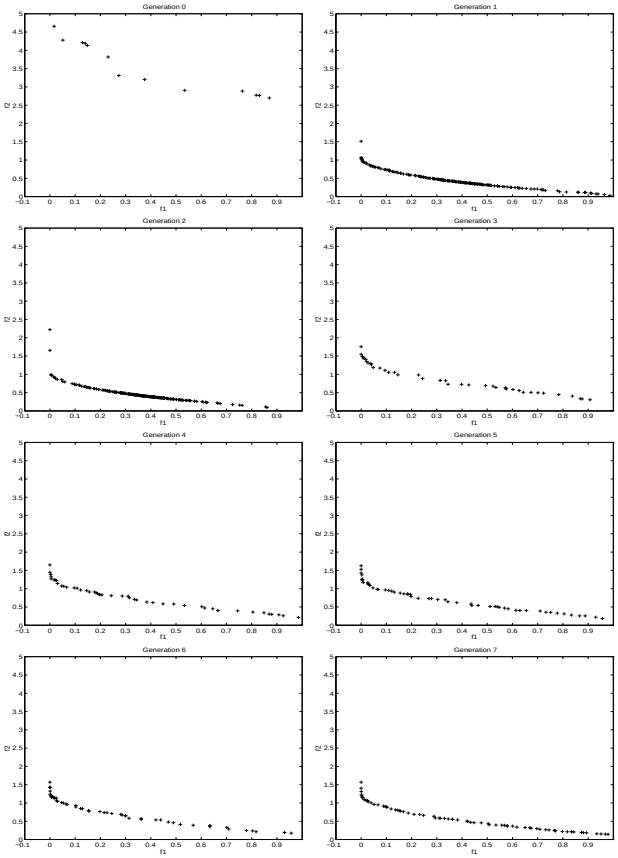

Fig. 1. The dynamics of SPDE in the first eight generations for one of the runs for Function P1.

C.2 The Parameters' Effect on SPDE

In this section, we investigate the performance of SPDE by varying the maximum number of objective evaluations and population size. To do this, we vary the maximum number of objective evaluations between 10,000 to 100,000 with a step of 10,000 and the population size between 100 to 250 with a step of 50. To have a proper comparison, we always compare against SPDE without mutation with the different parameters and PAES20. Figures 2, 3, 4, and 5 present the results for the four test problems.

A population size of 100 seems to have a good performance compared to other population sizes. For problem 1, in Figure 2, with 10,000 objective evaluations, SPDE completely dominates PAES20. For problem 2, in Figure 3, SPDE dominates PAES20 soon after 20,000 objective evaluations. With problems 3 and 4, in Figures 4 and 5, SPDE dominates PAES20 with the smallest number of objective evaluations of 10,000. An increase in the number of objective evaluations does not change the

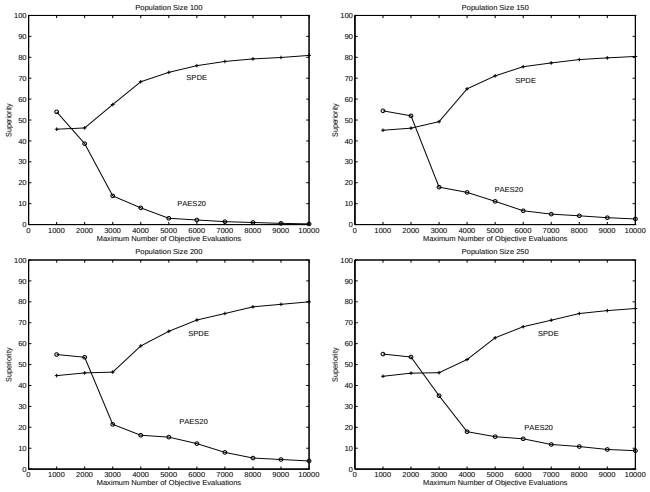

Fig. 2. The effect of maximum number of objective evaluations and population size on SPDE as compared to PAES20 for problem P1.

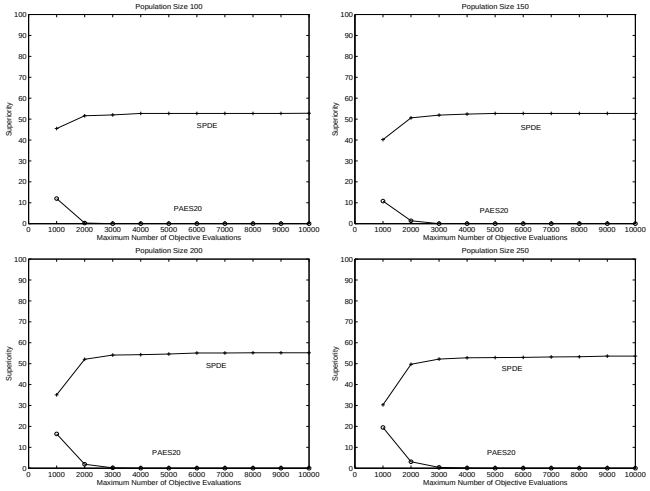

Fig. 3. The effect of maximum number of objective evaluations and population size on SPDE as compared to PAES20 for problem P2.

percentage of the pareto frontier where SPDE dominates PAES20. When the population size increases, it takes more objective evaluations to generate the same effect. This is a general trend with the four problems and therefore may suggest that longer search is more important than large populations.

V. Conclusions and Future Research

In this paper, a self–adaptive version of the pareto differential evolution algorithm was introduced for multiobjective optimization problems. The approach self–

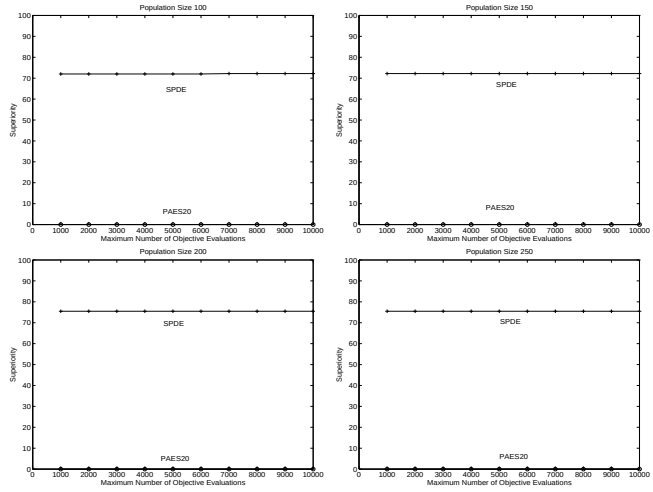

Fig. 4. The effect of maximum number of objective evaluations and population size on SPDE as compared to PAES20 for problem P3.

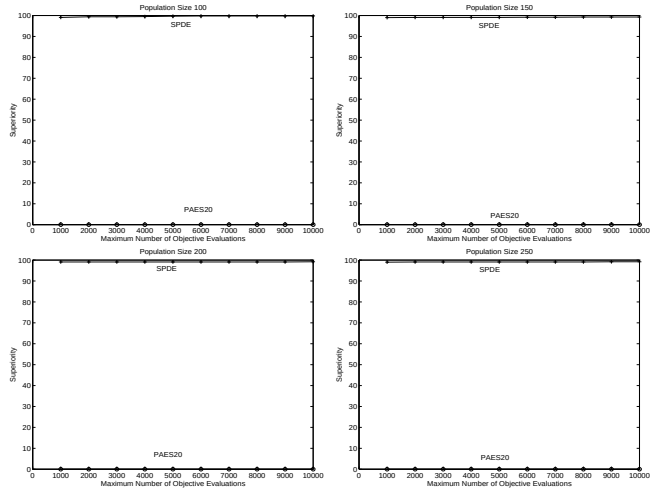

Fig. 5. The effect of maximum number of objective evaluations and population size on SPDE as compared to PAES20 for problem P4.

adapts the crossover and mutation rates. We tested the approach on four benchmark problems, where it outperformed a number of the state–of–the–art approaches in the literature. We also examined the effect of population size and maximum number of objective evaluations. It was found that the increase in population size did not have a large effect on the performance on our test cases. In addition, the approach did not require large number of objective evaluations to outperform the other algorithms.

For future work, we intend to further examine the SPDE algorithm and improve its performance for multiobjective optimization problems. It is our intention to minimize the user load with regard to the parameter's choices to minimum to make the approach more attractive for decision makers in real life domains.

Acknowledgements

The author 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, Eckart Zitzler for making the results of eight algorithms available on his webpage and Carlos Coello for maintaining the EMO web site.

References

[1] H.A. Abbass, R. Sarkar, and C. Newton. A pareto differential evolution approach to vector optimisation problems. *The IEEE Congress on Evolutionary Computation, Seoul, Korea*, pages 971–978, 2001.

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

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.