Solving Rotated Multi-objective Optimization Problems Using Differential Evolution

Antony W. Iorio and Xiaodong Li

School of Computer Science and Information Technology,
Royal Melbourne Institute of Technology University,
Melbourne, Vic. 3001, Australia
{iantony, xiaodong}@cs.rmit.edu.au
http://goanna.cs.rmit.edu.au/~xiaodong/ecml/

Abstract. This paper demonstrates that the self-adaptive technique of Differential Evolution (DE) can be simply used for solving a multi-objective optimization problem where parameters are interdependent. The real-coded crossover and mutation rates within the NSGA-II have been replaced with a simple Differential Evolution scheme, and results are reported on a rotated problem which has presented difficulties using existing Multi-objective Genetic Algorithms. The Differential Evolution variant of the NSGA-II has demonstrated rotational invariance and superior performance over the NSGA-II on this problem.

1 Introduction

Traditional genetic algorithms that use low mutation rates and fixed step sizes have significant trouble with problems with interdependent relationships between decision variables, but are perfectly suited to many of the test functions currently used in the evaluation of genetic algorithms [1]. These test functions are typically linearly separable and can be decomposed into simpler independent problems. Unfortunately, many real-world problems are not linearly separable, although linear approximations may sometimes be possible between decision variables.

Interdependencies between variables can be introduced into a real-coded functional problem by rotating the coordinate system of the test function. A rotated problem is not amenable to the directionless step-sizes and low mutation rates that Genetic Algorithms typically use. Although the NSGA-II is a very robust multi-objective optimization algorithm it suffers from the same limitations as traditional Genetic Algorithms on these problems.

Previous work has reported on the poor performance of a number of MOEAs, including the NSGA-II, on a rotated problem [2]. Rotated problems typically require correlated self-adapting mutation step sizes in order to make timely progress in optimization. In contrast, Differential Evolution has previously demonstrated rotationally invariant behaviour in the single objective domain [3]. This provides motivation to further demonstrate the worth of DE as a technique for addressing rotated multi-objective optimization problems. Our survey of the literature found that no work has explicitly demonstrated rotationally invariant

G.I. Webb and Xinghuo Yu (Eds.): AI 2004, LNAI 3339, pp. 861–872, 2004.

behaviour in multi-objective problems, therefore we propose a simple alteration to the NSGA-II to make it rotationally invariant. The mutation and crossover operators within the NSGA-II have been replaced with a Differential Evolution algorithm for generating candidate solutions. Differential Evolution has all the desired properties necessary to handle complex problems with interdependencies between input parameters, without the implementation complexity and computation cost of some self-adaptive Evolutionary Strategies [3].

A number of experiments have been conducted on a uni-modal rotated problem from the literature [2]. We have found that integrating Differential Evolution within the NSGA-II achieves rotational invariance on this problem.

The following section provides a brief introduction to the important concepts of Multi-objective Optimization, Differential Evolution, and Rotated Problems. Section 3 discusses the proposed model the Non-dominated Sorting Differential Evolution (NSDE) which integrates Differential Evolution with the NSGA-II. Section 4 outlines the performance metrics used in this study. Section 5 describes the experiments that were conducted, followed by the parameter settings and discussion of results in Section 6 and 7. The outcomes of this work and some possible future directions are outlined in Section 8.

2 Background

2.1 Multi-objective Optimization

Multi-objective optimization deals with optimization problems which are formulated with some or possibly all of the objective functions in conflict with each other. Such problems can be formulated as a vector of objective functions $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), .., f_n(\mathbf{x}))$ subject to a vector of input parameters $\mathbf{x} = (x_1, x_2, ..., x_m)$, where n is the number of objectives, and m is the number of parameters. A solution $\mathbf{x}$ dominates a solution $\mathbf{y}$ if objective function $f_i(\mathbf{x})$ is no worse than objective function $f_i(\mathbf{y})$ for all n objectives and there exists some objective j such that $f_j(\mathbf{x})$ is better than $f_j(\mathbf{y})$. The non-dominated solutions in a population are those solutions which are not dominated by any other individual in the population. Multiobjective evolutionary optimization is typically concerned with finding a diverse range of solutions close to the Pareto-optimal front, which is the globally non-dominated region of the objective space.

A number of evolutionary multiobjective algorithms have been developed since the late 80s, and the NSGA-II [2] is typically regarded as one of the best.

The criteria for evaluating the performance of a multiobjective evolutionary algorithm are different from those for assessing the performance of single objective algorithms. Generally, a multi-objective optimization produces a set of solutions. Therefore, we need to assess the final solution set in terms of uniform coverage of the Pareto-optimal front, closeness to the front, and spread across the front. In section 4 we will outline in more detail the performance metrics used in this study.

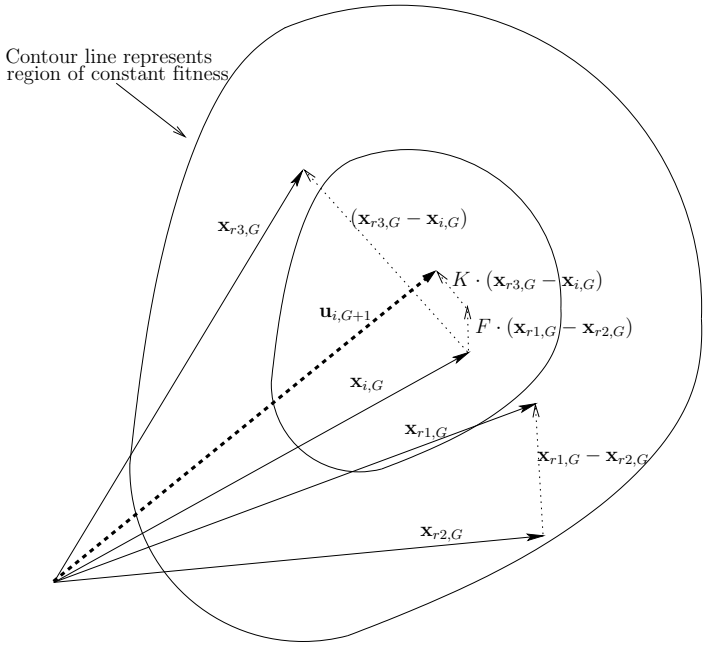

Fig. 1. The above figure shows the vector addition and subtraction necessary to generate a new candidate solution in *DE/current-to-rand/1*

2.2 Differential Evolution

Differential Evolution is a population-based direct-search algorithm for global optimization [4]. It has demonstrated its robustness and power in a variety of applications, such as neural network learning [5], IIR-filter design [6], and the optimization of aerodynamic shapes [7]. It has a number of important characteristics which make it attractive as a global optimization technique, and the reader is referred to [3] for an excellent introduction to DE which covers this in more detail. The primary property of Differential Evolution that will be the topic of study in this paper is rotational invariance.

Differential Evolution differs from other EAs in the mutation and recombination phase. Unlike stochastic techniques such as Genetic Algorithms and Evolutionary Strategies, where perturbation occurs in accordance with a random quantity, Differential Evolution uses weighted differences between solution vectors to perturb the population.

$$\begin{aligned} &\textit{randomly select } \; r_1, r_2, r_3 \in \{1, 2, ..., n\};\; r_1 \neq r_2 \neq r_3 \neq i \\ &\mathbf{u}_{i,G+1} = \mathbf{x}_{i,G} + K \cdot (\mathbf{x}_{r3,G} - \mathbf{x}_{i,G}) + F \cdot (\mathbf{x}_{r1,G} - \mathbf{x}_{r2,G}) \end{aligned} \tag{1}$$

The Differential Evolution variant used in this work is known as *DE/current-to-rand/1* (Equation 1), and is rotationally invariant [3]. The population of a

Differential Evolutionary Algorithm is typically randomly initialised within the initial parameter bounds. At each generation G, the population undergoes perturbation. Three unique individuals, or solution vectors denoted by $\mathbf{x}$, are randomly selected from the population. The coefficient K is responsible for the level of combination that occurs between $\mathbf{x}_{r3,G}$ and the current individual $\mathbf{x}_{i,G}$. The coefficient F is responsible for scaling the step size resulting from the vector subtraction $\mathbf{x}_{r1,G} - \mathbf{x}_{r2,G}$. Figure 1 details the relationship between the vectors responsible for the generation of a new candidate solution. Typically in the single-objective case, if the new individual $\mathbf{u}_{i,G+1}$,evaluates better than the currently selected individual $\mathbf{x}_{i,G}$, then the current individual is replaced with the new one. The algorithm iterates over i from 1 to n, where n is the size of the population.

2.3 Multi-objective Differential Evolution

Differential Evolution has also been applied to multi-objective problems. One of the first examples of this was to tune a fuzzy controller for the automatic operation of a train, although the cost function transformed the objectives of punctuality, comfort, and energy usage into the degenerate case of a single objective [8]. The Pareto Differential Evolutionary Algorithm (PDE) uses non-dominated solutions for reproduction, and places offspring back into the population if they dominate the current parent [9, 10]. This PDE was also extended into a variant with self-adaptive crossover and mutation [11]. Multi-objective DE has also been applied to minimize the error and the number of hidden units in neural network training. The resulting Pareto-front is the tradeoff between these two objectives [12]. The non-dominated sorting, ranking, and elitism techniques utilised in the NSGA-II have also been incorporated into a Differential Evolution method [13]. Another approach involving Pareto-based evaluation has also been applied to an Enterprise Planning problem with the two objectives of cycle time and cost [14], and also compared with the Strength-Pareto Evolutionary Algorithm [15].

2.4 Rotated Problems

A function can be rotated through one or more planes in the parameter space, where the number of planes is determined by the dimensionality of the problem. A problem with D dimensions in the parameter space has $D(D-1)/2$ possible planes of rotation. A problem rotated through all possible parameter space planes means that every variable has interdependencies with every other.

In order to generate a rotated problem, each solution vector $\mathbf{x}$ is multiplied by a rotation matrix $\mathbf{M}$, and the result is assigned to $\mathbf{y}$ (Equation 2). The new vector is then evaluated on each of the objective functions.

Figure 2 demonstrates the effect of rotation on the multi-objective problem in Equation 2 with two input parameters. The shapes of the functions stay the same, but their orientations change.

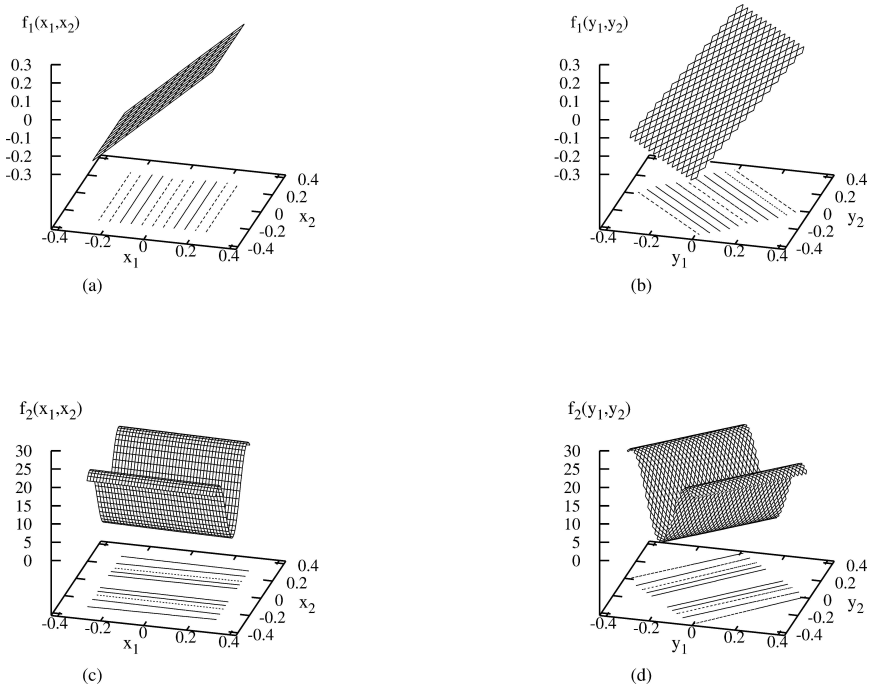

Fig. 2. The above figure shows the effect of a 45-degree rotation on the x_1x_2 plane on function f_1 and f_2. Before rotation, the functions are aligned with the coordinate system ((a) and (c)), and after rotation they are not ((b) and (d))

$$\begin{aligned} &minimize \quad f_1(\mathbf{y}) = y_1 \; and \; f_2(\mathbf{y}) = g(\mathbf{y})exp(-y_1/g(\mathbf{y})) \\ &where \;\; g(\mathbf{y}) = 1 + 10(D-1) + \sum_{i=2}^{D} \left[y_i^2 - 10\cos(4\pi y_i)\right] \\ &and \;\; \mathbf{y} = \mathbf{M}\mathbf{x}, \; -0.3 \leq x_i \leq 0.3, \; for \; i = 1, 2, ..., D. \end{aligned} \tag{2}$$

It is apparent from the contour plots in Figure 2 that before rotation the functions are aligned with the coordinate system. In which case, it is possible to make progress in the search by perturbing the parameters x_1 and x_2 independently. With rotated problems, significant progress in the search can only proceed by making simultaneous progress across all parameters within a solution vector. Consider Figure 3, where the elliptical contour represents a region of constant fitness. The point $\mathbf{v}$ can be perturbed along both the x_1 and x_2 axes, and any location along the dashed line will be an improvement over any point along the contour, assuming that the global optimum is centered on the coordinate axis. After rotation, progress from perturbing the same rotated point $\mathbf{v}'$ will be lower. This is because the interval of potential improvement for each of the decision variables is reduced, meaning that the search will progress more slowly when the parameters are only perturbed independently of each other. Another aspect

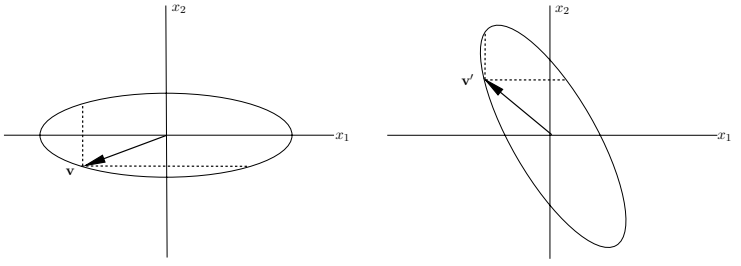

Fig. 3. The above figure demonstrates how rotation can reduce the interval of possible improvement. When the function is aligned with the coordinate axes, the improvement interval (dashed line) is larger than when the function is rotated away from the coordinate axes. The ellipse represents the region of constant fitness. Vector $\mathbf{v}$ and $\mathbf{v}'$ represent the same point in the search space before and after rotation respectively

of rotated problems is that points can easily be trapped along a valley line in the search space and can only make progress with simultaneous improvements over all input parameters (Figure 4). The point $\mathbf{v}$ can easily be perturbed in the x_1 axis to find the global minimum in the center of the coordinate system. The same point $\mathbf{v}'$ after rotation is still on the valley, but now it can not progress to a point of improved fitness by only moving along the direction of the coordinate axes (dashed line) because any such perturbation will be to a point of lower fitness in the search space. Typically the valley can be found easily, but the search often becomes trapped at this location. Only a simulatenous improvement in all parameters will result in the discovery of fitter solutions. On these types of problems, the small mutation rates frequently used in Genetic Algorithms are known to be even less efficient than a random search [1]. Self-adaptation has been relatively successful at solving this sort of problem using Evolutionary Strategies, but it requires the learning of appropriate correlated mutation step sizes and

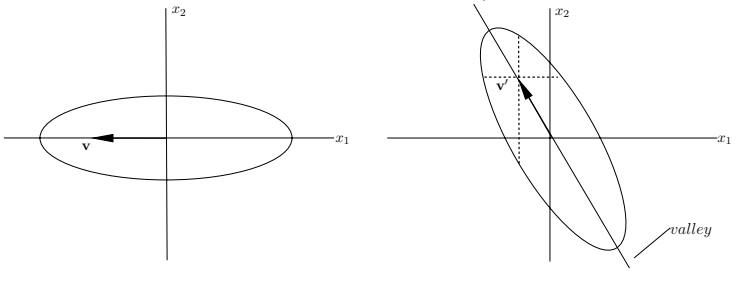

Fig. 4. The above figure demonstrates how rotation can trap points along the valley. If the point $\mathbf{v}'$ moves anywhere along the dashed lines it will be towards a point in the parameter space of lower fitness. Vector $\mathbf{v}$ and $\mathbf{v}'$ represent the same point in the search space before and after rotation respectively

it can be rather computationally expensive when D becomes large [3]. Differential Evolution is an attractive solution to this problem because of its ability to adapt to the fitness landscape through the correlation of mutation step sizes by sampling multiple times the difference between randomly selected solution vectors.

3 NSDE: A Simple Modification to the NSGA-II

The NSGA-II algorithm uses elitism and a diversity preserving mechanism. N offspring are created from a parent population of size N. The combined population of size $2N$ is sorted into separate non-domination levels. Individuals are selected from this combined population to be inserted into the new population, based on their non-domination level. If there are more individuals in the last front than there are slots remaining in the new population of size N, a diversity preserving mechanism is used. Individuals from this last front are placed in the new population based on their contribution to diversity in the population. The algorithm then iterates until some termination condition is met. The NSGA-II uses a real-coded crossover and mutation operator but in the multi-objective implementation of *DE/current-to-rand/1*, NSDE (Non-dominated Sorting Differential Evolution), these mutation and recombination operators were not used, and were replaced with Differential Evolution. In the single objective implementation of the Differential Evolution, if the new candidate $\mathbf{u}_{i,G+1}$ evaluates better than the current individual $\mathbf{x}_{i,G}$, the current individual is replaced with the new individual. In the multi-objective implementation this is not possible because we do not know which individual is better until all candidates are sorted together and assigned to a non-domination level. Therefore, $\mathbf{u}_{i,G+1}$ is first added to the new candidate offspring population. New candidates are generated using *DE/current-to-rand/1* until the candidate offspring population is filled up to size N. The new individuals are then evaluated on the objective functions, and then subjected to the combined non-dominated sorting described above. For further details regarding the implementation of the NSGA-II, the reader is referred to [2].

4 Performance Metrics

A number of performance metrics have been proposed for the purposes of comparing multiobjective optimization algorithms [16]. An analysis of different performance assessment techniques is provided in [17].

We use the following performance metrics introduced by Zitzler [18]. These metrics are frequently employed in the literature and we use them here to facilitate the comparison of the results with others. Secondly, they do not attempt to combine coverage, convergence or spread measures into a scalar, but provide these measures as distinct results. This assists any evaluation of the algorithms in relation to these measures. The * designates we have used the objective space

variant of these metrics only. Because metrics alone are probably insufficient to assess the performance of a multiobjective optimization algorithm [16], we have also provided plots of the non-dominated solutions (Figure 5).

$$\mathcal{M}_1^*(Y') := \frac{1}{|Y'|} \sum_{p' \in Y'} \mathbf{min}\{||\mathbf{p}' - \bar{\mathbf{p}}||^*; \bar{\mathbf{p}} \in \bar{\mathbf{Y}}\} \tag{3}$$

$$\mathcal{M}_2^*(Y') := \frac{1}{|Y' - 1|} \sum_{p' \in Y'} |\{\mathbf{q}' \in \mathbf{Y}'; ||\mathbf{p}' - \mathbf{q}'||^* > \sigma^*\}| \tag{4}$$

$$\mathcal{M}_3^*(Y') := \sqrt{\sum_{i=1}^{n} \mathbf{max}\{||\mathbf{p}'_\mathbf{i} - \mathbf{q}'_\mathbf{i}||^*; \mathbf{p}', \mathbf{q}' \in \mathbf{Y}'\}} \tag{5}$$

Y' is the set of objective vectors corresponding to the non-dominated solutions found, and $\bar{Y}$ is a set of uniform Pareto-optimal objective vectors. $\mathcal{M}_1^*(Y')$ provides a measure of convergence to the Pareto-optimal front by giving the average distance from Y' to $\bar{Y}$. The smaller the value of $\mathcal{M}_1^*(Y')$ the better, as the distance between Y' to $\bar{Y}$ should be minimised. This metric is useful when the true Pareto-front is known, although other metrics for measuring convergence to the front are appropriate when this is not the case [16].

$\mathcal{M}_2^*(Y')$ describes how well the solutions in Y' cover the front. $\mathcal{M}_2^*(Y')$ should produce a value between $[0, |Y'|]$ as it estimates the number of niches in Y' based on the niche neighbourhood size of σ^*. A niche neighbourhood size, $\sigma^* > 0$, is used in Equation 4 to calculate the distribution of the non-dominated solutions. Objective vectors outside the niche range are counted for each objective vector p' in Y'. The higher the value for $\mathcal{M}_2^*(Y')$ the better the coverage is across the front, according to σ^*.

$\mathcal{M}_3^*(Y')$ measures the spread of Y', which provides an indication of how well the search has spread to the extremes of the Pareto-optimal front. Large values of $\mathcal{M}_3^*(Y')$ are desired.

None of these metrics can be considered individually. For example, a good convergence of the population towards the Pareto-front may also have a poor coverage across the front, or vice versa.

5 Experiments

Experiments were conducted on the rotated problem described in section 2.4. The dimensionality of the parameter space was 10, resulting in 45 possible planes of rotation. Rotations were performed on each plane, introducing non-linear dependencies between all parameters. In order to demonstrate the rotational invariance of the NSDE on the problem, we performed experiments with 0 degrees of rotation (no parameter interactions) up to 45 degrees of rotation, at 5 degree intervals. Each experiment was run 30 times, for a total of 800 generations (80,000 evaluations) for each run. For comparative purposes the same experiments were performed with the NSGA-II as well. Results are presented in Figure 5, and Table 1.

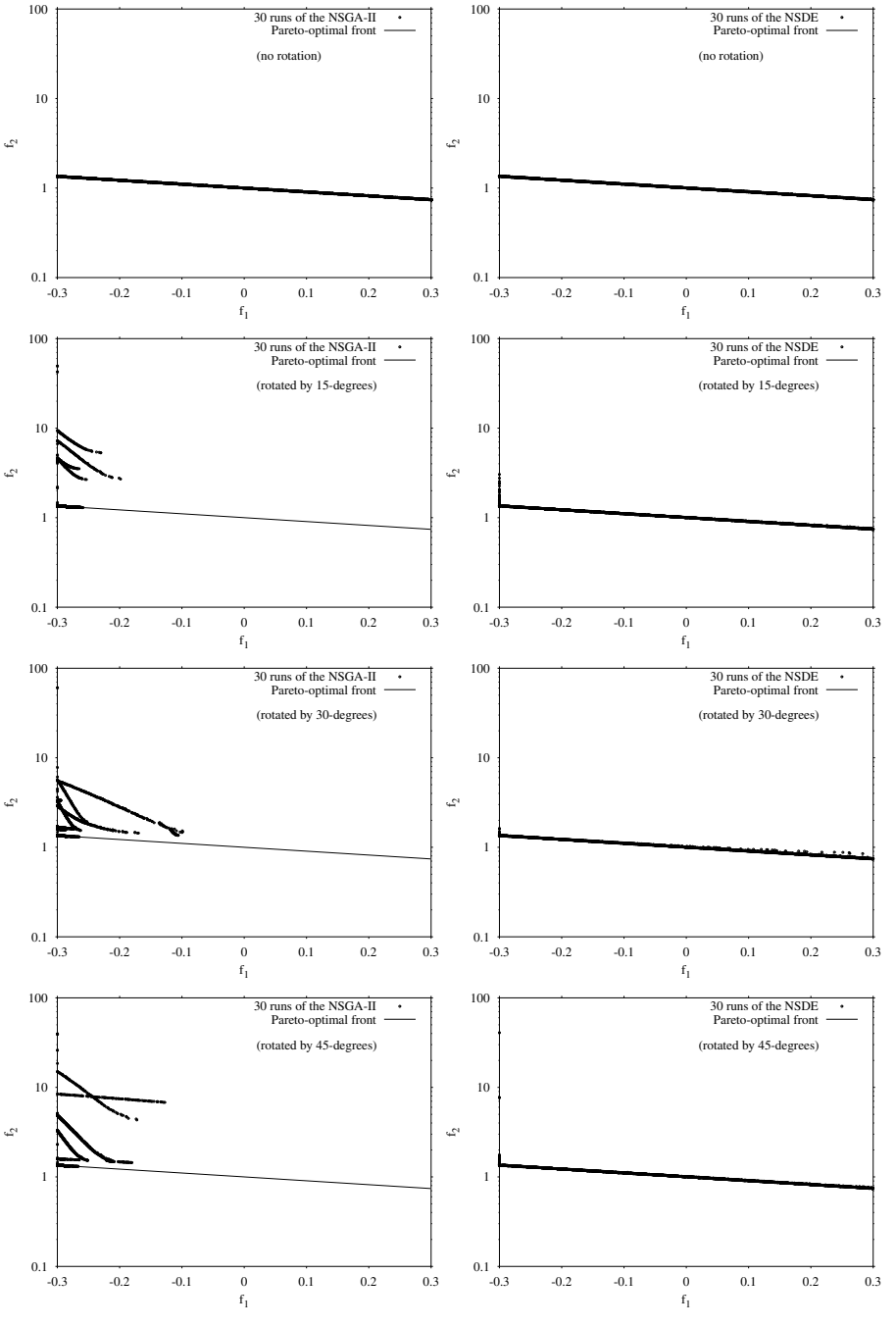

Fig. 5. Each of the left and right plots respectively show 30 runs of the NSGA-II and the NSDE algorithm on the rotated problem, with successively increasing degrees of rotation on all planes

Table 1. $\mathcal{M}_1^*$, $\mathcal{M}_2^*$, $\mathcal{M}_3^*$, and the number of evaluations (averaged over 30 runs). R_d represents the rotated problem where d is the degree of rotation on each plane

Metric	Algorithm	R_0	R_5	R_{10}	R_{15}	R_{20}
$\mathcal{M}_1^*$	NSGA-II	6.26E-04 ±7.55E-05	2.42E-02 ±9.31E-02	1.49E+00 ±3.07E+00	1.14E+00 ±1.62E+00	3.49E-01 ±6.85E-01
	NSDE	2.22E-03 ±1.97E-04	3.76E-03 ±5.22E-03	5.60E-03 ±1.18E-02	2.95E-01 ±7.51E-01	1.10E-01 ±4.16E-01
$\mathcal{M}_2^*$	NSGA-II	9.86E+01 ±7.00E-02	8.56E+01 ±2.46E+00	6.87E+01 ±2.42E+01	5.35E+01 ±2.35E+01	6.41E+01 ±1.98E+01
	NSDE	9.85E+01 ±5.49E-02	9.85E+01 ±1.28E-01	9.85E+01 ±2.14E-01	9.86E+01 ±2.22E-01	9.85E+01 ±4.68E-01
$\mathcal{M}_3^*$	NSGA-II	1.10E+00 ±5.48E-06	5.44E-01 ±3.33E-01	6.47E-01 ±7.94E-01	5.37E-01 ±6.27E-01	8.66E-01 ±1.52E+00
	NSDE	1.10E+00 ±7.48E-04	1.10E+00 ±1.79E-02	1.11E+00 ±3.68E-02	1.12E+00 ±8.01E-02	1.17E+00 ±3.56E-01

Metric	Algorithm	R_{25}	R_{30}	R_{35}	R_{40}	R_{45}
$\mathcal{M}_1^*$	NSGA-II	3.71E-01 ±5.47E-01	5.18E-01 ±7.91E-01	8.97E-01 ±2.08E+00	8.17E-01 ±1.79E+00	1.01E+00 ±1.87E+00
	NSDE	2.36E-03 ±4.41E-19	1.38E+00 ±1.07E+00	3.86E-03 ±5.06E-03	2.29E-02 ±6.26E-02	5.82E-01 ±9.78E-01
$\mathcal{M}_2^*$	NSGA-II	6.94E+01 ±2.66E+01	5.22E+01 ±3.61E+01	5.24E+01 ±3.42E+01	4.60E+01 ±3.39E+01	5.11E+01 ±3.75E+01
	NSDE	9.86E+01 ±5.22E-01	9.85E+01 ±1.19E-01	9.85E+01 ±3.13E-01	9.83E+01 ±1.36E+00	9.86E+01 ±2.00E-01
$\mathcal{M}_3^*$	NSGA-II	9.34E-01 ±1.22E+00	8.39E-01 ±1.47E+00	5.61E-01 ±9.51E-01	7.95E-01 ±1.75E+00	1.43E+00 ±2.32E+00
	NSDE	1.32E+00 ±5.81E-01	1.10E+00 ±3.14E-02	1.09E+00 ±5.03E-02	1.11E+00 ±1.50E-01	1.28E+00 ±9.62E-01

6 Parameter Settings

A population size of 100 was used for both the NSDE and the NSGA-II. A crossover rate of 0.9 and mutation rate of 0.1 were used with the NSGA-II. η_c and η_m are parameters within the NSGA-II which control the distribution of the crossover and mutation probabilities and were assigned values of 10 and 50 respectively. The choice of the NSGA-II parameters is the same as the parameter values previously used on this rotated problem in other work [2]. For the NSDE, F was set to 0.8 and K was set to 0.4. Suggestions from the literature helped guide our choice of parameter values for the NSDE [3]. The niche neighbourhood size σ^* described in section 4, for the metric $\mathcal{M}_2^*(Y')$, was set to 0.01.

7 Results and Discussion

From Table 1 it is apparent that the NSDE maintains a significantly better convergence ($\mathcal{M}_1^*$), coverage ($\mathcal{M}_2^*$), and spread ($\mathcal{M}_3^*$) than the NSGA-II. Figure 5

contains plots of 30 runs of the final non-dominated set after 80,000 evaluations. These figures further demonstrate that the NSDE consistently converged closely to the Pareto-optimal front, independently of the degree of rotation.

The only difference between the NSDE and the NSGA-II is in the method of generating new individuals. NSDE uses the step sizes of Differential Evolution which are adaptively adjusted to the fitness landscape. In contrast, the NSGA-II uses real-coded crossover and mutation operators. It is obvious that the cause of the poor performance by the NSGA-II on the rotated problem is because the perturbation of variables through mutation and crossover is not correlated. We have demonstrated that Differential Evolution can provide rotationally invariant behaviour on a multi-objective optimization problem, and we expect this should be true for other rotated problems as well. It is significant that such striking results were obtained from such a simple variation of the NSGA-II.

8 Conclusion

Outside of Evolutionary Strategies, Differential Evolution is currently one of a few techniques for solving multi-objective optimization problems with interdependencies between variables. The striking results on the single test problem we have investigated in this preliminary study suggest that further work is worthwhile. Currently we are investigating a number of even harder rotated problems, incorporating some of the features of existing test functions, such as multi-modality, non-uniformity, and discontinuities.

References

1. Salomon, R.: Re-evaluating Genetic Algorithm Performance Under Coordinate Rotation of Benchmark Functions: A Survey of Some Theoretical and Practical Aspects of Genetic Algorithms. In: Bio Systems, Vol. 39, No. 3. (1996) 263–278
2. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, In: IEEE Trans. Evol. Comput., Vol. 6, No. 2. (2002) 182–197
3. Price, K V.: An Introduction to Differential Evolution. In: Corne, D., Dorigo, M., and Glover, F. (eds.): New Ideas in Optimization. McGraw-Hill, London (UK) (1999) 79–108
4. Price, K. V.: Differential evolution: a fast and simple numerical optimizer. In: Smith, M., Lee, M., Keller, J., Yen., J. (eds.): Biennial Conference of the North American Fuzzy Information Processing Society, NAFIPS. IEEE Press, New York (1996) 524–527
5. Ilonen, J., Kamarainen, J.-K., Lampinen, J.: Differential Evolution Training Algorithm for Feed-Forward Neural Networks. In: Neural Processing Letters Vol. 7, No. 1 (2003) 93-105
6. Storn, R.: Differential evolution design of an IIR-filter. In: Proceedings of IEEE International Conference on Evolutionary Computation ICEC'96. IEEE Press, New York (1996) 268-273

7. Rogalsky, T., Derksen, R.W. and Kocabiyik, S.: Differential Evolution in Aerodynamic Optimization. In: Proceedings of the 46th Annual Conference of the Canadian Aeronautics and Space Institute. (1999) 29–36
8. Chang, C. S. and Xu, D. Y.: Differential Evolution of Fuzzy Automatic Train Operation for Mass Rapid Transit System. In: IEEE Proceedings of Electric Power Applications Vol. 147, No. 3 (2000) 206-212
9. Abbass, H. A., Sarker, R. and Newton, C.: PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems. In: Proceedings of the 2001 Congress on Evolutionary Computation (CEC'2001) Vol. 2 (2001) 971-978
10. Abbass, H. A. and Sarker, R.: The Pareto Differential Evolution Algorithm. In: International Journal on Artificial Intelligence Tools Vol. 11, No. 4 (2002) 531-552
11. Abbass, H. A.: The Self-Adaptive Pareto Differential Evolution Algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002) Vol. 1, IEEE Press, (2002) 831-836
12. Abbass, H. A.: A Memetic Pareto Evolutionary Approach to Artificial Neural Networks. In: Proceedings of the Australian Joint Conference on Artificial Intelligence, Adelaide, Australia, Lecture Notes in Artificial Intelligence Vol. 2256, Springer-Verlag, (2001) 1-12
13. Madavan, N. K.: Multiobjective Optimization Using a Pareto Differential Evolution Approach. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002) Vol. 2, IEEE Press, (2002) 1145-1150
14. Xue, F.: Multi-Objective Differential Evolution and its Application to Enterprise Planning. In: Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA'03) Vol. 3, IEEE Press, (2003) 3535-3541
15. Xue, F., Sanderson, A. C. and Graves, R. J.: Pareto-based Multi-objective Differential Evolution. In: Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003) Vol. 2, IEEE Press, (2003) 862-869
16. Okabe, T., Jin, Y. and Sendhoff B.: A Critical Survey of Performance Indicies for Multi-Objective Optimisation. In: Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003) Vol. 2, IEEE Press, (2003) 878-885
17. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., Fonseca, V. G.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. In: IEEE Trans. Evol. Comput., Vol. 2, No. 2. (2003) 117–132
18. Zitzler, E., Deb, K. and Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, **8**(2):173-195, April (2000).