International Journal of Computational Intelligence and Applications
Vol. 7, No. 2 (2008) 1–38

Imperial College Press
www.icpress.co.uk

ROTATED PROBLEMS AND ROTATIONALLY INVARIANT CROSSOVER IN EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

ANTONY IORIO* and XIAODONG LI†

School of Computer Science and IT
RMIT University, GPO Box 2476v
Melbourne VIC 3001, Australia
**iantony@cs.rmit.edu.au*
†*xiaodong@cs.rmit.edu.au*

Received
Revised

Problems that are not aligned with the coordinate system can present difficulties to many optimization algorithms, including evolutionary algorithms, by trapping the search on a ridge. The ridge problem in single-objective optimization is understood, but until now little work has been done on understanding this issue in the multi-objective domain. Multi-objective problems with parameter interactions present difficulties to an optimization algorithm, which are not present in the single-objective domain. In this work, we have explained the nature of these difficulties, and investigated the behavior of the NSGA-II, which has difficulties with problems not aligned with the principle coordinate system. This study has investigated Simplex Crossover (SPX), Unimodal Normally Distributed Crossover (UNDX), Parent-Centric Crossover (PCX), and Differential Evolution (DE), as possible alternatives to the Simulated Binary Crossover (SBX) operator within the NSGA-II, on problems exhibiting parameter interactions through a rotation of the coordinate system. An analysis of these operators on three rotated bi-objective test problems, and a four- and eight-objective problem is provided. New observations on the behavior of rotationally invariant crossover operators in the multi-objective problem domain have been reported.

Keywords: Evolutionary computation; evolutionary multi-objective optimization; parameter interactions.

1. Introduction

Traditional genetic algorithms that use low mutation rates and fixed step sizes have significant trouble with problems that have interdependent relationships between decision variables, but are perfectly suited to many of the test functions currently used in the evaluation of genetic algorithms.[1] Test functions that are typically employed are 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.

This issue has previously been explored in the single-objective optimization domain,[1] where interactions between decision variables were introduced by rotating

the coordinate system of test functions. A rotated problem poses serious problems to the directionless step sizes and low mutation rates that genetic algorithms typically use,[1] and demonstrates a weakness in nonvector-wise approaches to optimization.

Parameter interactions that hamper optimization algorithms are not only an issue for single-objective optimization, but they also exist in multi-objective problems. In a scenario where there may be two or more conflicting objectives, typically there will exist a set of optimal solutions that are trade-offs against each objective. Evolutionary algorithms are particularly well suited to solving this type of problem because they deal with a population of individuals. The issue of parameter interactions in such multi-objective problems has recently drawn increasing attention, as the poor performance of the NSGA-II[2] was reported on multi-objective problems exhibiting parameter interactions.[2,40]

Although the NSGA-II is a very robust multi-objective optimization algorithm it suffers from similar limitations as the canonical genetic algorithm does in the single-objective problem domain. The reason for this limitation is the nature of the crossover technique employed. NSGA-II uses a crossover technique called Simulated Binary Crossover (SBX),[3,4] combined with a uniform crossover where half the time parameters of an offspring solution are replaced with parameters from a parent solution. This crossover technique can only search effectively along the principle coordinate axes. As a result, finding optimal solutions becomes extremely difficult when the decision space dimension increases. This is because a simultaneous improvement on each decision space parameter is required in order to find a more optimal solution. Problems that are rotated exhibit this characteristic and typically require correlated self-adapting mutation step sizes in order to make timely progress in searching for optimal solutions.[1]

There are of course a number of recombination techniques that are invariant under a rotation of a coordinate system, such as Differential Evolution (DE), which has previously demonstrated rotationally invariant behavior in the single-objective domain.[6] Other rotationally invariant techniques for generating offspring include Simplex Crossover[13] (SPX), Parent-Centric Crossover[18] (PCX), and Unimodal Normal Distribution Crossover[14,15,17] (UNDX-m) have also demonstrated rotationally invariant behavior on single-objective test problems. These crossover techniques have the necessary characteristics to handle problems with interdependencies between decision variables, without the computation cost of self-adaptive Evolutionary Strategies.[6] This provides the motivation to study the worth of these multi-parent crossover techniques on rotated multi-objective optimization problems.

This paper builds upon the work in Ref. 7, where a simple multi-objective DE scheme demonstrated dramatically improved performance on a rotated test problem. Further work proposed a number of rotated test problems,[8] which were also subsequently tested on a DE technique which made use of directional information in the multi-objective problem domain.[10] Some preliminary work was also reported on rotated test problems.[9] In the work presented here, we have investigated a

number of simple alterations to the NSGA-II in order to study the behaviors of the aforementioned rotationally invariant crossover techniques.

In this paper, some of the important background concepts related to this study will be introduced first; Sec. 2 introduces multi-objective optimization, and how multi-objective problems are specified. Following this, Sec. 3 details the effect of parameter interactions on single-objective and multi-objective genetic algorithms. In Sec. 3, a unimodal multi-objective problem, which was originally proposed in Ref. 2, is used to facilitate an explanation of the difficulties associated with parameter interactions in multi-objective optimization. The construction of rotated multi-objective test problems is described in Sec. 4, where two new rotated multi-objective problems are also introduced: a problem with a nonuniform mapping between the objective and decision spaces and a problem with a discontinuous Pareto-optimal front. In Sec. 5, a new problem based on the hyper-ellipsoid function is introduced, which is rotatable in the decision space and scalable to multiple objectives. This problem is used to study the effect of parameter interactions as the objective space dimension increases. Each of the recombination operators used in this study is described in Sec. 6, as well as how they are used within the NSGA-II framework. The performance assessment criteria are introduced in Sec. 7, and the experiment settings with a description of the methodology employed are described in Sec. 8. A discussion of the results of each algorithm variant on each test problem is presented in Sec. 9. Finally, some concluding remarks and new observations about the performance of multi-objective evolutionary algorithms are provided in Sec. 10.

2. Multi-objective Optimization

Multi-objective optimization deals with optimization problems that 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 $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}))$ subject to a vector of input parameters $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, where M is the number of objectives, and N is the number of parameters. Although an objective can be either a minimization or maximization objective, within this study all objectives are minimization objectives.

The solution to a multi-objective problem is typically a set of trade-off solutions, where the first solution may be better on objective f_1 but worse on objective f_2, and the second solution may be worse on objective f_1 but better on objective f_2. Multi-objective evolutionary optimization is typically concerned with finding a diverse range of solutions in such a set, close to the Pareto-optimal front, which is the globally nondominated region of the objective space.

The criteria for evaluating the performance of a multi-objective evolutionary algorithm are different from those for assessing the performance of single-objective algorithms. Generally, a multi-objective optimization produces a set of solutions. For the purposes of this study these sets will be assessed with a metric described in Sec. 7, which can be inversely applied to measure both the convergence to the Pareto-optimal front and the diversity of the nondominated solution set.

3. Epistasis from Rotated Problems

Although epistatic interactions can be introduced in other representations, such as binary, for the purposes of this study, we are only considering real-valued representations because all of the crossover operators being investigated, except for DE, are designed to work with real-valued representations.

In the following section, we explain the effect of rotation on a simple ellipsoid minimization problem, $f(x_1, x_2) = x_1^2 + a_0 x_2^2$, with a global minimum located at $x_1 = 0$ and $x_2 = 0$ (Fig. 1). This function is linearly separable, aligned with the principle coordinate axes, and can be solved as two independent problems by decomposing it into the function $f_1 = x_1^2$ and $f_2 = a_0 x_2^2$. In other words, an optimization algorithm needs only to perturb the variables x_1 and x_2 independently in order to find the global optimum. If a separable problem, like the ellipsoid problem, is rotated away from the principal coordinate axes, the decision variables become dependent, and the function becomes linearly inseparable. After rotation, the ellipsoid function becomes $f_1 = x_1^2 + a_1 x_1 x_2 + a_0 x_2^2$, introducing parameter interactions through the term $a_1 x_1 x_2$. With rotated problems, significant progress in the search can only proceed by making simultaneous progress across all parameters within a solution vector.

Consider Fig. 1, 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 of rotated problems is that points can easily be trapped along a valley (or ridge) line in the search space and can only make progress with simultaneous improvements over all input

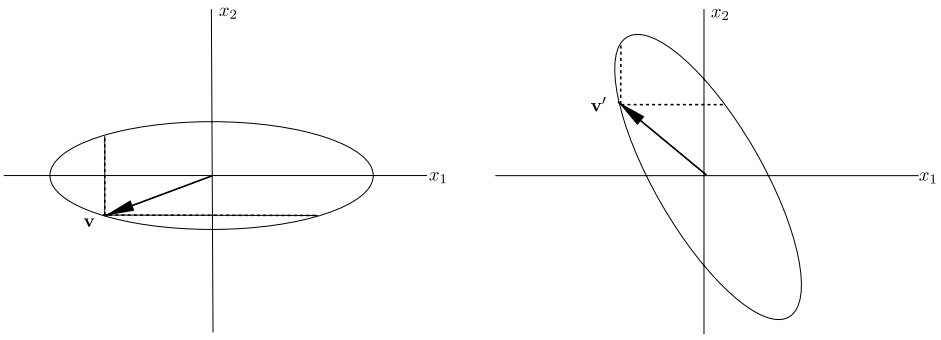

Fig. 1. Rotation can reduce the interval of possible improvement. When the function is aligned with the coordinate axes, the improvement interval (dashed line) is larger than when the function is rotated away from the coordinate axes. The ellipse represents the region of constant fitness. Vector $\mathbf{v}$ and $\mathbf{v}'$ represent the same point in the search space before and after rotation, respectively.

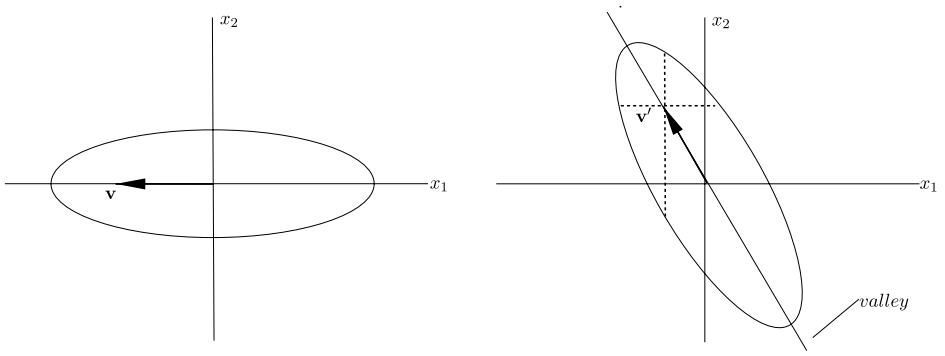

Fig. 2. Rotation can trap points along the valley. If the point $\mathbf{v}'$ moves anywhere along the dashed lines it will be toward a point in the parameter space of worse fitness. Vector $\mathbf{v}$ and $\mathbf{v}'$ represent the same point in the search space before and after rotation, respectively.

parameters (Fig. 2). 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 cannot 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 worse fitness in the search space. Typically the valley can be found easily, but the search often becomes trapped at this location. Only a simultaneous 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 it can be rather computationally expensive when the decision space dimension becomes large.[6]

The situation is not quite as simple with respect to the behavior resulting from nonrotationally invariant crossover operators on a multi-objective rotated problem. The difference between rotated single-objective and multi-objective problems can be demonstrated with Problem (1) in Fig. 3, which was first proposed in Ref. 2, and constructed using the framework in Ref. 42. The framework used to construct this problem will be elaborated upon in Sec. 4:

$$\left.\begin{aligned} f_1(\mathbf{y}) &= y_1 \\ f_2(\mathbf{y}) &= g(\mathbf{y})h(f_1(\mathbf{y}), g(\mathbf{y})) \\ h(f_1(\mathbf{y}), g(\mathbf{y})) &= \exp\left(\frac{-f(\mathbf{y})}{g(\mathbf{y})}\right) \\ g(\mathbf{y}) &= 1 + 10(N-1) + \sum_{i=2}^{N}\left[y_i^2 - 10\cos(4\pi y_i)\right] \\ \mathbf{y} &= \mathbf{O}\mathbf{x},\ -0.3 \le x_i \le 0.3, \quad for\ i = 1, 2, \ldots, N \end{aligned}\right\} \tag{1}$$

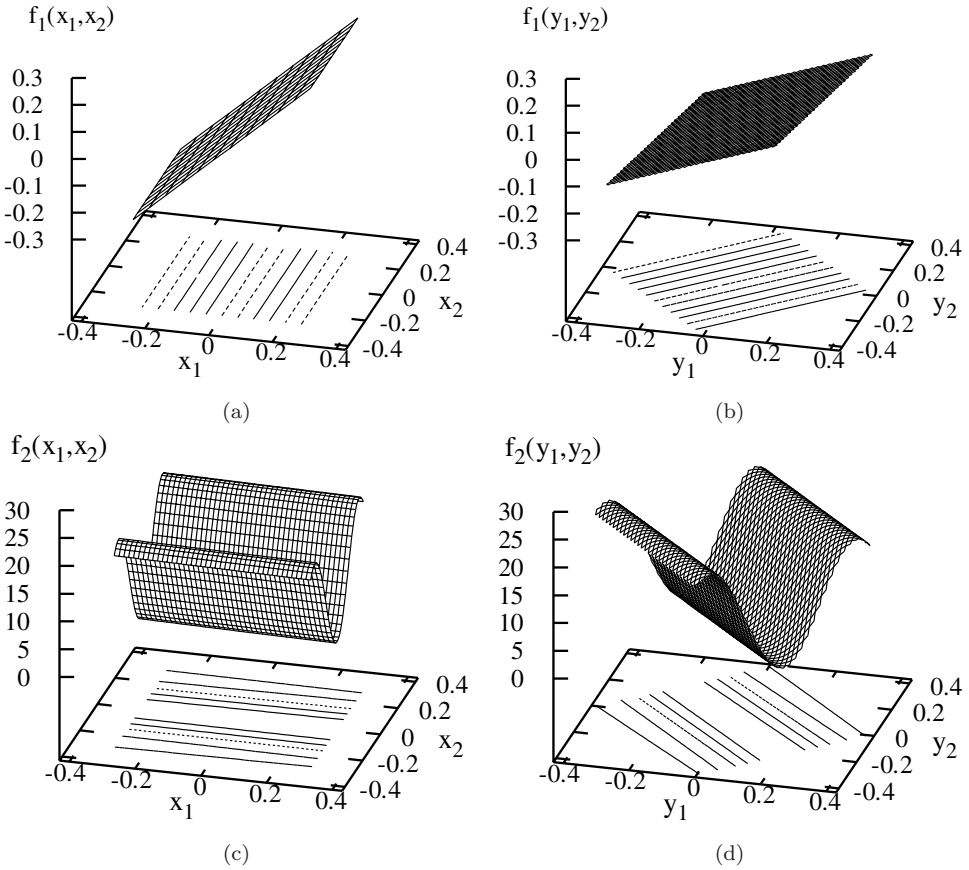

Fig. 3. The effect of a 45° rotation on the x_1x_2 plane of Problem (1). Before rotation, the functions are aligned with the coordinate system ((a) and (c)), and after rotation they are not aligned with the coordinate system ((b) and (d)).

This problem will facilitate our understanding of the effect of rotation on multi-objective problems where nondominated solution sets are sought by an optimization algorithm. The situation is analogous to the single-objective domain, where an optimization algorithm with independent perturbations on each decision variable will have trouble finding more optimal solutions.

Figure 3 shows Problem (1) with a two-dimensional decision space. This problem is characterized by a slightly inclined valley in objective f_2. Objective f_1 is a plane with a sloping gradient in an opposing direction to the incline of objective f_2. The Pareto-optimal set is represented by a line segment bisecting the decision space in objective f_2 and f_1, respectively. The decision space vector $\mathbf{x}$ is subject to a rotation matrix $\mathbf{O}$, resulting in the rotated vector $\mathbf{y}$.

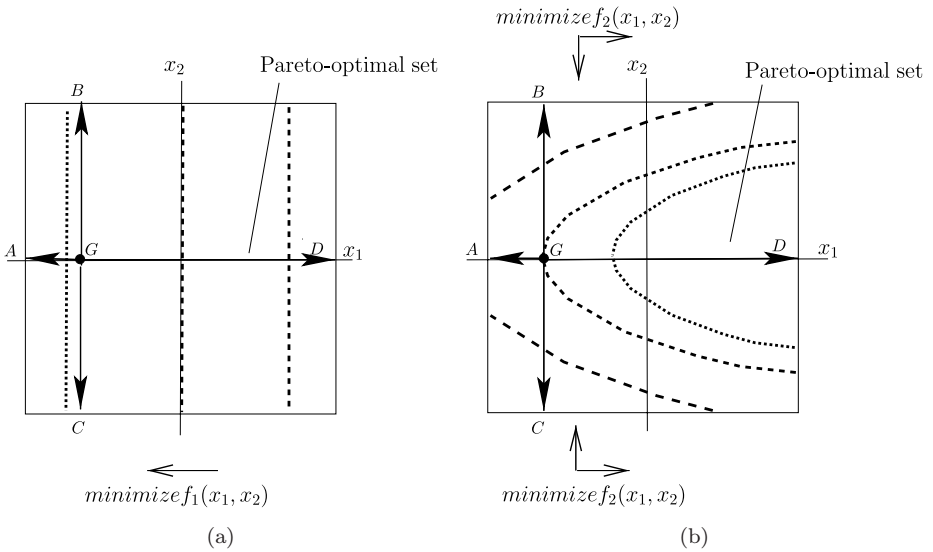

Fig. 4. The contour plot of nonrotated problem (1) in (a) represents function f_1, and the contour plot in (b) represents objective function f_2. The dashed lines represent regions of constant value with respect to the objective function evaluation. Smaller dash sizes represent lower evaluations on the objective functions.

Consider the contour plot of a nonrotated version of this problem in Fig. 4, where a point, G, is a member of the Pareto-optimal set. If the point G is perturbed in the direction of $\mathbf{GA}$, it is toward points that evaluate lower with respect to objective f_1 (Fig. 4(a)) and higher with respect to objective f_2 (Fig. 4(b)). If it is perturbed in the direction of $\mathbf{GD}$, it will be toward points that evaluate higher with respect to objective f_1 and lower with respect to objective f_2. Such perturbations are with respect to the parameter x_1 only, and this is the only such perturbation required in order to discover other Pareto-optimal solutions, which are located on the line segment bisecting the contour plots for objectives f_1 and f_2. It is apparent that such a Pareto-optimal solution can easily be perturbed toward other Pareto-optimal solutions when the problem is aligned with the principle coordinate axes. However, after the problem has been rotated, it becomes more difficult to find Pareto-optimal solutions through independent perturbations of individuals. Consider Fig. 5, where point G' represents the point G after rotation. Through independent perturbations of decision space parameters, the point G can perturb to other nondominated solutions in the direction of $\mathbf{G'A'}, \mathbf{G'B'}$, $\mathbf{G'C'}$, and $\mathbf{G'D'}$. A perturbation in the direction of $\mathbf{G'A'}$ leads to points that evaluate lower on objective f_1, but higher on objective f_2. This is similarly true for perturbations in the direction of $\mathbf{G'B'}$. A perturbation in the direction of $\mathbf{G'C'}$ leads to points that evaluate higher on objective f_1, but lower on objective f_2. This is also true for perturbations in the direction of $\mathbf{G'D'}$. Unfortunately, each of these perturbations leads to nondominated solutions that skew away from the Pareto-optimal set. The

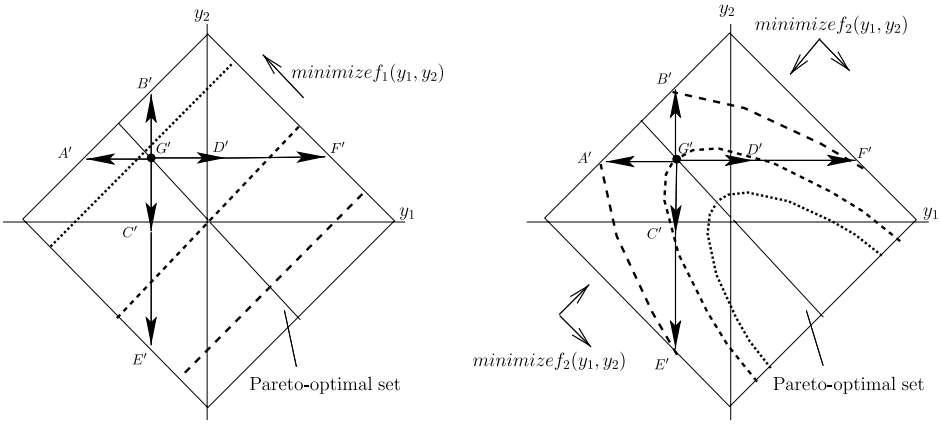

Fig. 5. The contour plot of rotated problem (1) in (a) represents function f_1, and the contour plot in (b) represents objective function f_2.

situation becomes even worse if the perturbation extends to $\mathbf{C'E'}$ or $\mathbf{D'F'}$, because individuals in this region evaluate higher with respect to objective f_1 and objective f_2, and are dominated by the point at G', as a result. In actuality, the problem in Fig. 3 that this analysis is based on has an extremely small region relative to the feasible space, where nondominated solutions can be located in the direction of $\mathbf{G'C'}$ and $\mathbf{G'D'}$. Nondominated solutions in these directions only become increasingly likely as the orientation of the problem approaches alignment with the principle coordinate axes. As the orientation of the Pareto-optimal front aligns with the axis y_1, the line vector $\mathbf{G'C'}$ extends further and more nondominated solutions can be discovered in this region more easily. Secondly, such nondominated solutions will be close to the Pareto-optimal set. This is similarly true for the line vector $\mathbf{G'D'}$, as the Pareto-optimal set aligns with the axis y_2. In other words, a rotation of the problem that results in the Pareto-optimal set not being aligned with any principle coordinate axis makes it difficult to discover other Pareto-optimal solutions when only independent perturbations of decision variables can occur.

In the presence of only independent perturbations, there is a tendency for points to be discovered in the direction of lower f_1 evaluations and higher f_2 evaluations, pushing the nondominated solution set away from the Pareto-optimal set and degrading the search over time. As a result of this behavior, the search can become trapped in the Pareto-optimal region and fail to find more nondominated solutions in the Pareto-optimal set. Progress in covering the Pareto-optimal front becomes extremely slow. This effect was also apparent in Refs. 7 and 2, where the NSGA-II produced poor coverage of the Pareto-optimal front on the rotated unimodal multi-objective problem. Any multi-objective optimization algorithm, which is not rotationally invariant, will exhibit such behavior on a problem with similar characteristics.

Furthermore, the way that Problem (1) is structured causes clustering of ranked solutions, as a result of nondominated sorting if NSGA-II is used. This can exacerbate the problem, through the loss of coverage of the Pareto-optimal front early on in the search, making it difficult to regain a good coverage when the problem is rotated for reasons stated above. Of course, there may be problems that may be more or less sensitive to rotation, depending on the location of the Pareto-optimal front, the mapping between decision and objective space, the modalities present in the functions, and other aspects of the fitness landscape.

4. Constructing Problems with Parameter Interactions

As demonstrated in Sec. 3, interdependencies between variables can be introduced into a real-coded functional problem by rotating the coordinate system of a test problem. In order to construct a rotated problem, one must be careful that under rotation the problem can still evaluate to a meaningful result. Secondly, in order to achieve a completely unbiased assessment of an algorithm on a problem which is rotated, one must guarantee a *uniformly distributed random rotation*. This means that the orientation of a point resulting from a rotation is not biased toward any particular orientation. In other words, if one pictures a point on the surface of a unit hypersphere, a uniformly distributed random rotation around the centroid of the unit hypersphere shifts that point to another location on the surface of the unit hypersphere with uniform probability.

Figure 6 outlines the procedure for generating a random orthonormal basis, which is used to introduce parameter interdependencies into a problem by rotating the parameter vector. As stated previously, this is a linear transformation that does not change the fitness landscape of the problem domain. A normal distribution is used in this technique because it provides for the distribution of points centered

Step 1. Generate a random unit vector $\mathbf{o_1}$ by taking N independent normally distributed random variables with mean 0 and variance 1.
Step 2. $\mathbf{o_1} = \mathbf{o_1} / \parallel \mathbf{o_1} \parallel$
FOR $i = 2$ TO N
Step 3. Generate a random unit vector $\mathbf{o_i}$ by taking N independent normally distributed random variables with mean 0 and variance 1.
Step 4. Perform Gram-Schmidt Orthonormalization[5] of $\mathbf{o_i}$ with preceding $\mathbf{o_1}$ to $\mathbf{o_{i-1}}$
Step 5. $\mathbf{o_i} = \mathbf{o_i} / \parallel \mathbf{o_i} \parallel$
END FOR

Fig. 6. Algorithm for generating a random rotation matrix, $\mathbf{O} = [\mathbf{o_1}, \ldots, \mathbf{o_N}]^{\mathbf{T}}$, which distributes points uniformly on the surface of a hypersphere.

around the origin with perfect rotational symmetry. The normally distributed random variables are normalized so that each component of the rotated point maintains an equivalent distance from the axis of rotation. Therefore, a uniform distribution of points on the surface of a hypersphere is possible when the orthonormal basis[5] $\mathbf{o_1}, \ldots, \mathbf{o_N} \in \mathbb{R}^N$ is used to rotate a point in N-dimensional space. This technique also makes it possible to randomly and uniformly rotate a decision space vector, so that there is no bias for any particular coordinate axis.

It deserves to be noted that rotating the decision space is not the only technique for introducing parameter interactions to test problems. In Ref. 40, an approach was presented, which introduces parameter interactions between decision variables in a test problem, using a transformation matrix that performs shearing, scaling, and rotation. One of the limitations of this approach is that a shearing or scaling operator changes the fitness landscape; hence, the experimenter has to be particularly careful to maintain the desired characteristics of the fitness landscape in each objective. Care must be taken in any conclusions that are drawn, either as a result of parameter interactions introduced to the problem or as a result of a scaling or shearing of the original fitness landscape. Furthermore, the elements of the transformation matrix are sampled uniformly from -1 to 1. This introduces degenerate behavior if the matrix element that is sampled for the transformation matrix approaches 0. Under such circumstances, the fitness landscape can be flattened on a particular plane, removing any modalities that were present in the original problem.

This approach can be contrasted with the approach presented in this work, where we are only concerned with rotation, which is a linear transformation where only the orientation of the fitness landscape changes. The advantage of our approach is that parameter interactions can be introduced to a problem without altering the fitness landscape.[40] This enables one to clearly identify the effect of introducing parameter interactions to a problem, because the fitness landscape is unaltered.

Based on the approach described in Fig. 6 and the framework in Ref. 42, two test problems will be proposed. Before the problems are detailed, some points associated with the construction of test problems will be discussed. In the test problem framework used in this study, the g function is responsible for affecting convergence to the Pareto-optimal front, and the h function specifies the shape of the Pareto-optimal front. The Pareto-optimal set can be determined by setting the g function to 1.0, and evaluating f_1 over the range of feasible solutions. Diversity is also affected by the f_1 function.

In order to construct a problem with parameter interactions, the problem must have at least one nonlinear function. With this consideration, the rotatable test problems we have proposed in this paper will have at least one nonlinear function in at least one of the objective functions. Each of the test problems also sets f_1 and f_2 to large values if f_1 is outside the ranges specified in the problem descriptions. This is important because rotation may push a function evaluation outside the range desired by the experimenter.[42]

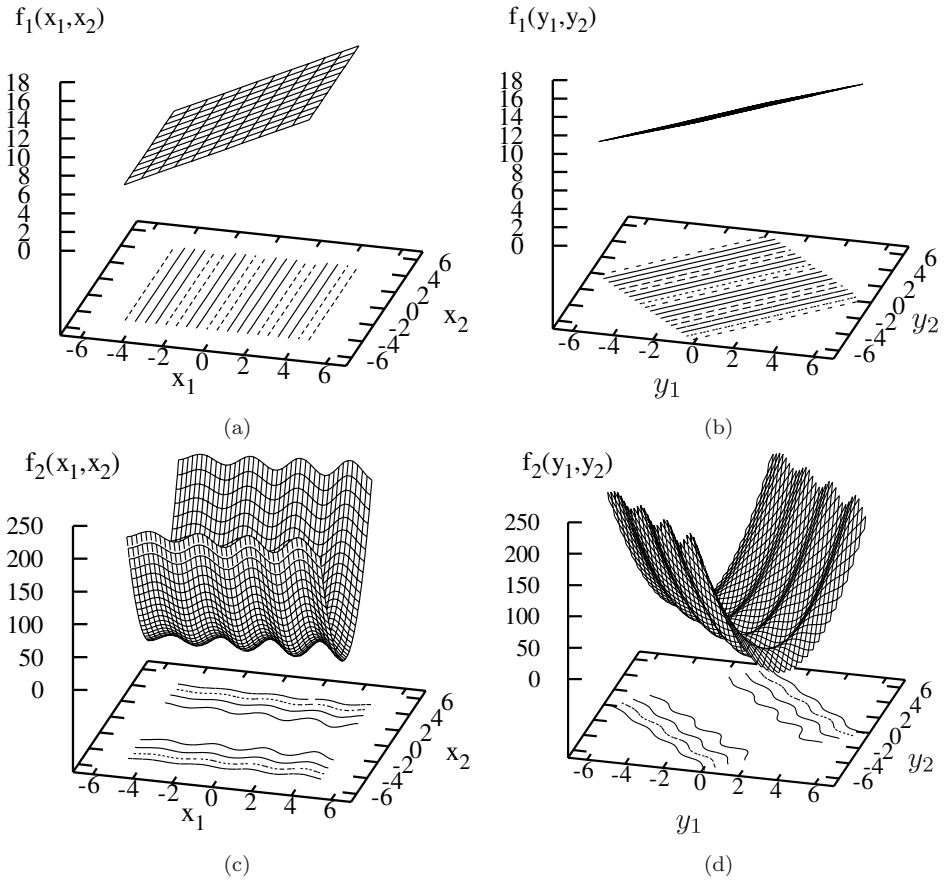

Fig. 7. The effect of a 45° rotation on the x_1x_2 plane on function f_1 and f_2 of Problem (2). Before rotation, the functions are aligned with the coordinate system ((a) and (c)), and after rotation they are not aligned with the coordinate system ((b) and (d)).

One must also be careful that under rotation the problem can still evaluate to a meaningful result. One should avoid situations where a rotation transformation results in decision variables, which take negative values and are then subjected to a square root function for instance. This can be achieved by avoiding functions, which would result in such a situation, or by offsetting the variable value within the function so a negative value never results. It should also be noted that the methodology for performing a rotation can be applied to other problems from the literature and is not limited to this framework. The only considerations that need to be addressed are related to making sure the evaluation of the rotated vector yields a result that can still be evaluated, and that the fitness landscape of the problem is unchanged as a result of a rotation transformation. In the test problems proposed

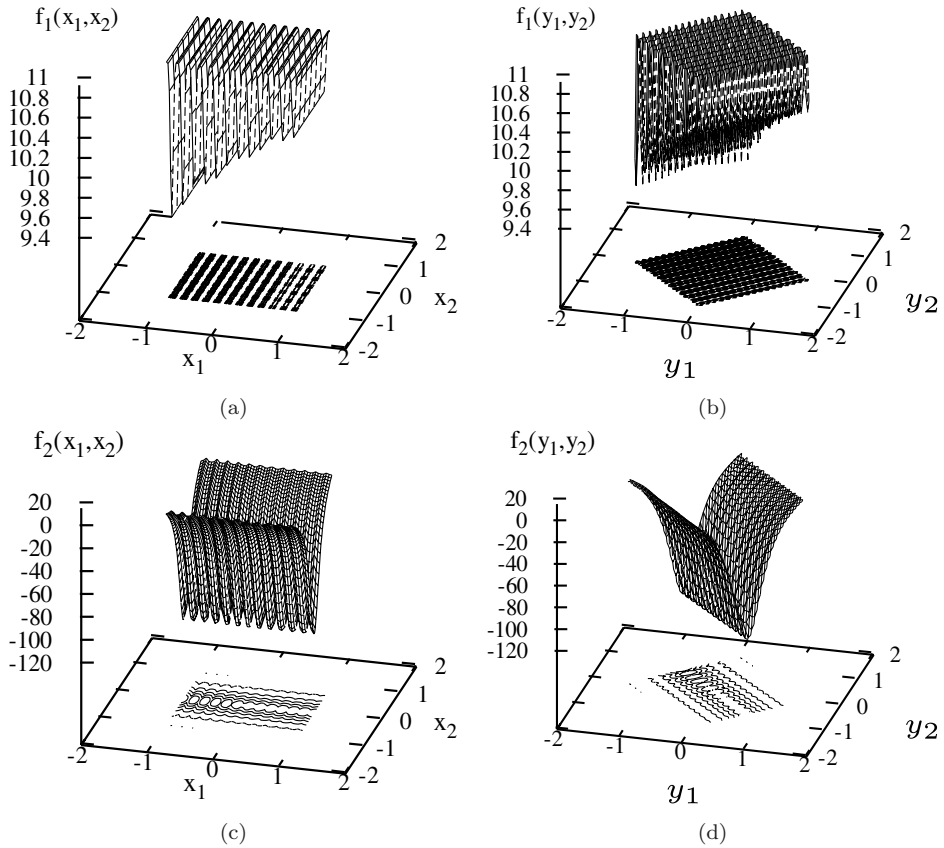

Fig. 8. The effect of a 45° rotation on the x_1x_2 plane on function f_1 and f_2 of Problem (3). Before rotation, the functions are aligned with the coordinate system ((a) and (c)), and after rotation they are not aligned with the coordinate system ((b) and (d)).

here, the decision space $\mathbf{x}$ is subjected to a rotation matrix $\mathbf{O}$ in order to generate a rotated vector $\mathbf{y}$ from $\mathbf{y} = \mathbf{Ox}$, and the fitness landscape is rotated as a result

$$\left.\begin{aligned} f_1(\mathbf{y}) &= y_1 + 10.0 \\ f_2(\mathbf{y}) &= g(\mathbf{y})h(f_1(\mathbf{y}), g(\mathbf{y})) \\ h(f_1(\mathbf{y}), g(\mathbf{y})) &= 1 + \frac{f_1(\mathbf{y})}{g(\mathbf{y})} + \left(\frac{f_1(\mathbf{y})}{g(\mathbf{y})}\right)\cos(0.8\pi f_1(\mathbf{y})) \\ g(\mathbf{y}) &= 1 + \frac{9}{N-1}\left[\sum_{i=2}^{N}\mathbf{y}^2\right] \\ \mathbf{y} &= \mathbf{Ox},\ -5 \le x_i \le 5 \quad \text{for } i = 1, 2, \ldots, N \end{aligned}\right\} \tag{2}$$

Based on the described approach, we have proposed two rotated problems, in addition to Problem (1), which was introduced in Sec. 3. Problem (2), which is plotted in a two-dimensional decision space in Fig. 7, has a Pareto-optimal front, which is not

continuous, as specified by the h function. f_1 is bounded to the range, $6 < f_1 < 16$, in order to guarantee that the values of f_1 do not go out of range during rotation. This problem presents a difficulty to an optimization algorithm, because such an algorithm has to locate a number of discontinuous Pareto-optimal fronts:

$$\left.\begin{aligned} f_1(\mathbf{y}) &= 1 - \exp(-0.5y_1)\sin^6(6\pi(y_1)) + 10.0 \\ f_2(\mathbf{y}) &= g(\mathbf{y})h(f_1(\mathbf{y}), g(\mathbf{y})) \\ h(f_1(\mathbf{y}), g(\mathbf{y})) &= 1 - \left(\frac{f(\mathbf{y})}{g(\mathbf{y})}\right)^2 \\ g(\mathbf{y}) &= 1 + \frac{9}{N-1}\left[\sum_{i=2}^{N}\mathbf{y}^2\right] \\ \mathbf{y} &= \mathbf{O}\mathbf{x},\ -1 \le x_i \le 1 \quad \text{for } i = 1, 2, \ldots, N \end{aligned}\right\} \tag{3}$$

Problem (3) is also plotted in a two-dimensional decision space in Fig. 8. In this problem, the decision space variables, which increment at a regular interval, evaluate with nonregular intervals in the objective space, making it hard to find a uniform distribution of points along the Pareto-optimal front. The density of solutions is lower toward lower f_1 values, making it difficult to find solutions in this region. f_1 is bounded to the range, $9.5 < f_1 < 10.5$, to guarantee that the values of f_1 do not go out of range during rotation. Plots of the objective space are not shown as the objective space is unchanged under a rotation of the decision space.

5. Rotated Problems with More Than Two Objectives

As described in Ref. 41, NSGA-II experiences difficulties when trying to optimize a large number of objectives because as the number of objectives increases, the number of nondominated solutions that are generated increases dramatically. Furthermore, the number of individuals in a population that are required to represent a Pareto-optimal front grows exponentially with respect to increasing objective space dimensions. NSGA-II has been found to optimize particular test problems in less than four dimensions, but when the objective space dimension was increased, NSGA-II could only find solutions far from the true Pareto-optimal front[23]:

$$\left.\begin{aligned} f_1(\mathbf{y}) &= 0.5(y_1 + 0.5)(y_2 + 0.5)\cdots(y_{M-1} + 0.5)(1 + g(\mathbf{y}_M)) \\ f_2(\mathbf{y}) &= 0.5(y_1 + 0.5)(y_2 + 0.5)\cdots(0.5 - y_{M-1})(1 + g(\mathbf{y}_M)) \\ &\vdots \qquad\qquad \vdots \\ f_{M-1}(\mathbf{y}) &= 0.5(y_1 + 0.5)(0.5 - y_2)(1 + g(\mathbf{y}_M)) \\ f_{M-1}(\mathbf{y}) &= 0.5(0.5 - y_1)(1 + g(\mathbf{y}_M)) \\ g(\mathbf{y}_M) &= \sum_{i=M}^{N} 2^{(i-M)}\left(y_i^2\right) \\ \mathbf{y} &= \mathbf{O}\mathbf{x},\ -0.5 \le x_i \le 0.5 \quad \text{for } i = 1, 2, \ldots, N \end{aligned}\right\} \tag{4}$$

These issues can further be compounded in the presence of parameter interactions between decision space variables. In order to study the effect of parameter interactions in problems with a large number of objectives, Problem (4) is proposed. This problem is similar to Problem DTLZ1,[21] the Pareto-optimal objective function values of Problem (4) lie on the linear hyperplane specified by $\sum_{m=1}^{M} \mathbf{f}^*{}_m = 0.5$. The total number of variables is $N = M + k - 1$. A value of $k = 5$ is used, which is the same k value used with DTLZ1. The g function specifies a hyper-ellipsoid function.[37] The coefficient term $2^{(i-M)}$ varies the eccentricity of the ellipsoid. This can make it difficult for an optimizer to adapt appropriate step sizes for each decision space dimension. When this function is rotated, it introduces the valley problem to an algorithm that can only perturb parameters independently. Interestingly, Problem (4) is unimodal and in that respect is simpler than the multimodal DTLZ1 problem, but can present significant challenges to an optimization procedure when rotated.

6. NSGA-II and Rotationally Invariant Crossover Operators

The NSGA-II is the multi-objective optimization algorithm used in this study. In this section, we will describe the procedure of this algorithm. Furthermore, each of the crossover variants used in this study, namely, SBX, UNDX-m, SPX, and PCX, will be described in detail.

In each of the variants of the NSGA-II, we have replaced the SBX crossover operation within the algorithm, with one of these crossover variants. In this study, the same polynomial mutation operator[11] that is employed in the NSGA-II is used with all variants, except for DE.

The NSGA-II algorithm uses elitism and a diversity preserving mechanism. *popsize* offspring are created from a parent population of *popsize*. The combined population of $2 * popsize$ is sorted into separate nondomination levels. Individuals are selected from this combined population to be inserted into the new population, based on their nondomination level. If there are more individuals in the last front than there are slots remaining in the new population of *popsize*, 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.

In the NSGA-II we do not know which individual is better until all candidates are sorted together and assigned to a nondomination level. Therefore, new individuals are added to the new candidate offspring population. New candidates are generated using a rotationally invariant crossover operator until the candidate offspring population is filled up to *popsize*. The new individuals are then evaluated on the objective functions and then subjected to the combined nondominated sorting mentioned previously. For further details regarding the implementation of the NSGA-II, the reader is referred to Ref. 2.

6.1. *SBX with uniform crossover*

The NSGA-II uses a SBX operator[4,11] with uniform crossover to generate offspring parameter values. The SBX operator takes two parents and produces two offspring, but does not have the property of rotational invariance because the correlation between the location of parents, and the location of offspring, which are generated, is lost under a rotation. The discrete crossover of variables also results in nonrotationally invariant behavior. For example, if an offspring vector has a parameter replaced by a parent parameter, as it might under some uniform crossover scheme, rotational invariance is destroyed.[6]

It has been shown that the SBX has a zero probability of generating some points in the space between two parents,[12] although in the new version of SBX implemented in the latest revision of NSGA-II (Sec. 10), this problem has been rectified somewhat by generating offspring in quadrants adjacent to the location of the parents, as well as around the parents (Fig. 9).

The new version of SBX, with a uniform crossover, uses the following procedure to generate offspring. μ_i is a uniform random number between 0 and 1, generated for each decision space parameter i. η_c is a nonnegative real number responsible for controlling the distribution of offspring that are created from the parents. A large value for η_c provides a higher probability of generating children near the parents, while a small value will generate offspring further from the parents. In step 1, if a uniform random number, $r_i^{(1)}$, between 0 and 1 is less than or equal to 0.5 we perform the crossover, else we copy the parents x_i to the children y_i. If the crossover is performed, the children are calculated from the parents and assigned to each parent according to the value of the uniform random number $r_i^{(2)}$.

Step 1: For each variable i, if $0 < r_i^{(1)} \le 0.5$ do the following

$$\beta_i = \begin{cases} (2\mu_i)^{\frac{1}{\eta_c+1}}, & 0 < \mu_i \le 0.5, \\ \left(\dfrac{1}{2(1-\mu_i)}\right)^{\frac{1}{\eta_c+1}}, & 0.5 < \mu_i \le 1, \end{cases}$$

$$\left.\begin{aligned} y_i^{(1)} &= 0.5((1-\beta_i)x_i^{(1)} + (1+\beta_i)x_i^{(2)}) \\ y_i^{(2)} &= 0.5((1+\beta_i)x_i^{(1)} + (1-\beta_i)x_i^{(2)}) \end{aligned}\right\} \quad \text{if } 0 < r_i^{(2)} \le 0.5,$$

$$\left.\begin{aligned} y_i^{(1)} &= 0.5((1+\beta_i)x_i^{(1)} + (1-\beta_i)x_i^{(2)}) \\ y_i^{(2)} &= 0.5((1-\beta_i)x_i^{(1)} + (1+\beta_i)x_i^{(2)}) \end{aligned}\right\} \quad \text{if } 0.5 < r_i^{(2)} \le 1,$$

else if $0.5 < r_i^{(1)} \le 1$,

$$\begin{aligned} y_i^{(1)} &= x_i^{(1)}, \\ y_i^{(2)} &= x_i^{(2)}. \end{aligned}$$

Step 2: Select two new parents, and repeat from step 1 until the desired number of offspring are generated.

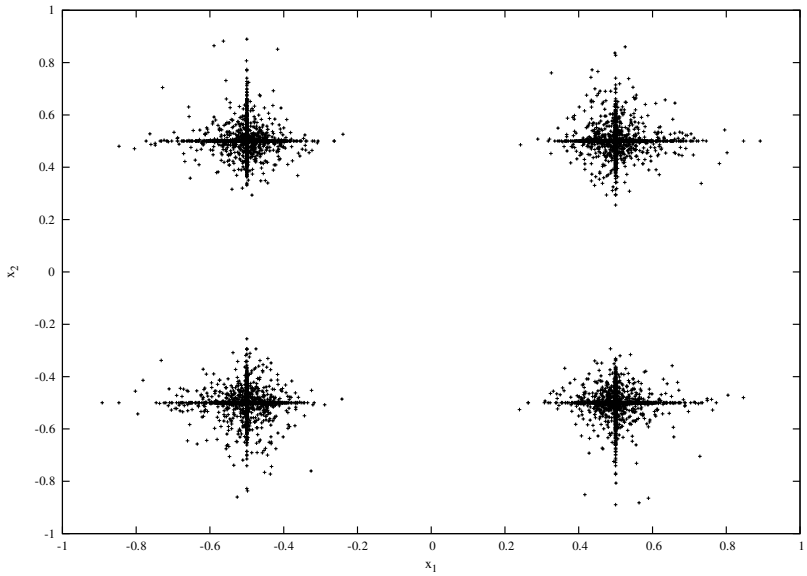

Fig. 9. Offspring resulting from parents $x_1 = (-0.5, 0.5)$ and $x_2 = (0.5, -0.5)$ for the new SBX operator with uniform crossover.

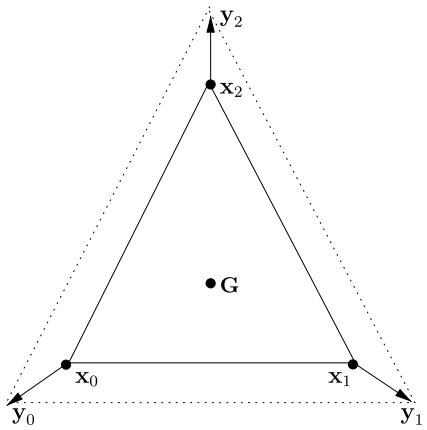

Fig. 10. In simplex crossover, new offspring are uniformly randomly generated within the expanded simplex.

6.2. *Simplex crossover (SPX)*

The SPX (Fig. 10), originally proposed in Ref. 13 has the primary feature of being independent of the coordinate system. In this approach, typically three parent individuals are chosen from the population. The simplex formed by these parents is expanded by a small degree specified by ϵ. Within this expanded simplex, one or more new individuals are sampled. One feature of this approach is that the majority of

sampling occurs uniformly within the region between the parents. Sampling outside this region is determined by the ϵ value. In a single-objective optimization procedure, as the population converges, the relative size of the expanded simplex will shrink as well. In a multi-objective evolutionary algorithm, a number of individuals are maintained in the population. As a result of this, as the population converges toward the Pareto-optimal set, one can expect the expanded simplex to select parents from the full range of nondominated individuals in the population. The following procedure occurs at each generation to generate the offspring.

Step 1: The simplex is first constructed by choosing $(N + 1)$ parents $\mathbf{x_i}$, where $i \in \{0, \ldots, N\}$, by randomly sampling from the population.

Step 2: Calculate the centroid of the simplex, $\mathbf{G} = \sum_{i=0}^{N} (\frac{\mathbf{x_i}}{N+1})$.

Step 3: Generate the expanded simplex points, $\mathbf{y_k} = \mathbf{G} + \epsilon(\mathbf{x_k} - \mathbf{G})$, where $k \in \{0, 1, \ldots, N\}$. The control parameter, ϵ, is responsible for the rate of expansion.

Step 4: Generate uniformly distributed random numbers, $\mathbf{r_k} = \text{rand}(0, 1)^{(\frac{1}{k+1})}$, where $k \in \{0, \ldots, N\}$.

Step 5: $\mathbf{C_k} = 0$ where $k = 0$. $\mathbf{C_k} = \mathbf{r_{k-1}}(\mathbf{y}_{k-1} - \mathbf{y}_k + \mathbf{C}_{k-1})$ where $k \in \{0, \ldots, N\}$.

Step 6: An offspring, $\mathbf{C}$, is generated by $\mathbf{y_N} + \mathbf{C_N}$.

Step 7: The offspring is mutated. (In our implementation, the mutation occurs by the polynomial mutation operator in the NSGA-II. Mutation is required by the SPX because otherwise there is insufficient exploration of the search space. A Gaussian mutation operator was also reportedly used in previous studies of the SPX[13].)

Step 8: Repeat from step 4 until the desired number of offspring is generated.

6.3. *Unimodal Normal Distribution Crossover (UNDX-m)*

The UNDX crossover[14] (Fig. 11) has demonstrated excellent performance in optimizing highly epistatic functions.[15] It has been applied to some difficult real-world problems such as design of optical lens systems.[16] A multi-parent variant of the UNDX was proposed, called UNDX-m.[14,17] The UNDX-m covers the search space more effectively by having a greater diversity of offspring generated, and it is this

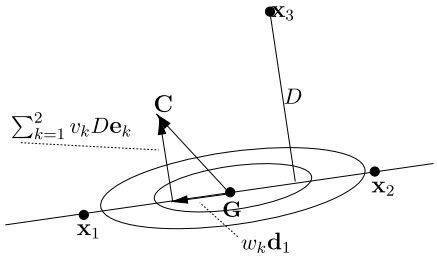

Fig. 11. In the UNDX, new offspring cluster around the centroid of the first $m + 1$ parents.

variant that we will be considering. In some respects, this approach is similar to the SPX, but unlike the simplex approach, offspring are sampled around a centroid of a number of parents. Offspring are generated around a centroid of the first $m+1$ parents, where m is a user specified variable. In the UNDX-m approach, the size of this sampled region and the extent of the region are controlled by two controlled parameters, namely σ_η and σ_ξ. The procedure for generating offspring using the UNDX-m approach is outlined as follows.

Step 1: Choose (m+2) parents $\mathbf{x_i}$, where $i \in \{1, \ldots, m+2\}$, by randomly sampling from the population.
Step 2: Calculate the centroid of the first $m+1$parents, $\mathbf{G} = \frac{1}{m+1}\sum_{i=1}^{m+1} \mathbf{x_i}$.
Step 3: Create the direction vectors, $\mathbf{d}_i = \mathbf{x}_i - \mathbf{G}$, where $i \in \{1, \ldots, m+2\}$.
Step 4: Let D be the length of the component of d_{m+2}, which is orthogonal to $d_1, \ldots, d_m$.
Step 5: Let $\mathbf{e}_1, \ldots, \mathbf{e}_{N-m}$ be the orthonormal bases of the subspace orthogonal to the subspace spanned by $d_1, \ldots, d_m$.
Step 6: An offspring, $\mathbf{C}$, is generated by $\mathbf{C} = \mathbf{G} + \sum_{k=1}^{m} w_k \mathbf{d}_k + \sum_{k=1}^{N-m} v_k D\mathbf{e}_k$, where w_k and v_k are normally distributed random variables $N(0, \sigma_\xi^2)$ and $N(0, \sigma_\eta^2)$, respectively.
Step 7: The offspring is mutated using the NSGA-II mutation operator.
Step 8: Repeat from step 6 until the desired number of offspring is generated.

6.4. *Parent-Centric Crossover (PCX)*

The PCX[18,22] (Fig. 12) is similar to the UNDX-m, but instead of distributing the offspring around the centroid of a number of parents, the offspring distribute in hyper-ellipsoids around the parents themselves. This contrasts with the UNDX-m and SPX approaches, which sample the majority of offspring between the parents.

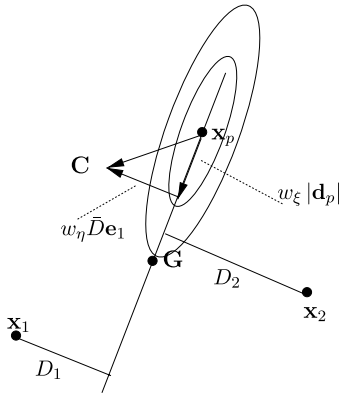

Fig. 12. In the PCX, new offspring cluster around a parent, x_p, which is selected with equal probability from the m chosen parents.

One potential advantage of parent-centric over centroid-centric is that sampling near parents can result in fitter offspring with similarities to the parents. Offspring are calculated from the PCX approach as follows.

Step 1: Choose m parents $\mathbf{x_i}$, where $i \in \{1, \ldots, m\}$, by randomly sampling from the population.
Step 2: Calculate the centroid of the parents from step 1, $\mathbf{G} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x_i}$.
Step 3: Choose one of the parents selected in step 1 with equal uniform probability, and assign to $\mathbf{x}_p$. The offspring will be centered around this parent.
Step 4: Create the direction vectors, $\mathbf{d}_p = \mathbf{x}_p - \mathbf{G}$.
Step 5: For the remaining $m-1$ parents, the perpendicular distances D_i to the line $\mathbf{d}_p$ are calculated and the average distance is assigned to $\bar{D}$.
Step 6: An offspring, $\mathbf{C}$, is generated by $\mathbf{C} = \mathbf{x}_p + w_\xi |\mathbf{d}_p| + \sum_{\substack{k=1 \\ k \neq p}}^{m} w_\eta \bar{D} \mathbf{e}_k$, where w_ξ and w_η are normally distributed random variables $N(0, \sigma_\xi^2)$ and $N(0, \sigma_\eta^2)$, respectively. Let each $\mathbf{e}_k$ be an orthonormal basis vector of the subspace orthogonal to the subspace spanned by d_p.
Step 7: The offspring is mutated using the NSGA-II mutation operator.
Step 8: Repeat from step 3 until the desired number of offspring are generated.

6.5. *Differential Evolution (DE)*

DE is a population-based direct-search algorithm for global optimization.[24] It has demonstrated its robustness and power in a variety of applications, such as neural network learning,[25] IIR-filter design,[26] and the optimization of aerodynamic shapes.[27] It has a number of important characteristics, which make it attractive as a global optimization technique, and the reader is referred to Ref. 6 for an excellent introduction to DE, which covers this in more detail. Some of these characteristics include the ability to self-adapt the step size; in other words as the population converges, the magnitude of the vector difference resulting from the DE calculation becomes smaller. This is a desirable characteristic because in any evolutionary algorithm exploration should occur earlier on, followed by a greater degree of exploitation of individuals. DE achieves this through its self-regulation of step size. Furthermore, the vector-wise sampling of DE allows the algorithm to be rotational invariant. The primary property of DE that is the topic of study in this paper is rotational invariance.

DE 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, DE uses weighted differences between solution vectors to perturb the population. At each generation, the following procedure is used to generate the offspring:

Step 1: $i = 0$.
Step 2: Randomly select $r_1, r_2, r_3 \in \{1, 2, \ldots, popsize\}$ such that $r_1 \neq r_2 \neq r_3 \neq i$ where i is the index of the currently selected individual in the population.

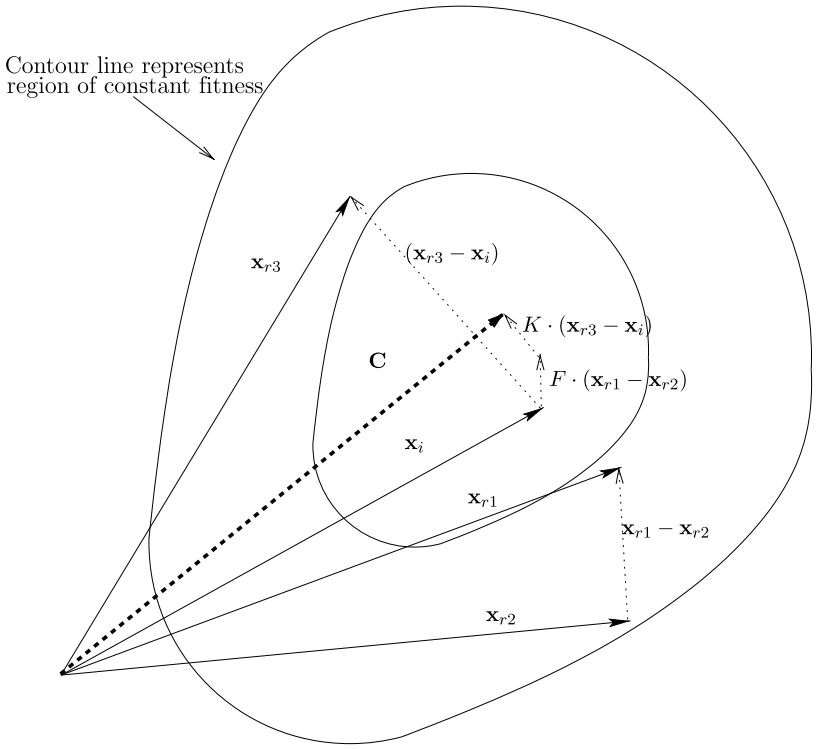

Fig. 13. Differential evolution.

Step 3: Generate an offspring from the selected parents and current individual, $\mathbf{C} = \mathbf{x}_i + K(\mathbf{x}_{r3} - x_i) + F(\mathbf{x}_{r1} - \mathbf{x}_{r2})$, where K and F are control parameters.

Step 4: $i = i + 1$.

Step 5: Repeat from step 2 if the desired number of offspring has not been reached and $i \neq popsize$.

The population of a differential evolutionary algorithm is typically randomly initialized within the initial parameter bounds. At each generation, 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 $\mathbf{x}_{r3}$ and the current individual $\mathbf{x}_i$. The coefficient F is responsible for scaling the step size resulting from the vector difference $\mathbf{x}_{r1} - \mathbf{x}_{r2}$. Figure 13 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 $\mathbf{C}$ evaluates better than the currently selected individual $\mathbf{x}_i$, then the current individual is replaced with the new one. The algorithm iterates over i from 1 to $popsize$.

DE 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.[28] The Pareto-differential evolutionary (PDE) algorithm uses nondominated solutions for reproduction and places offspring back into the population if they dominate the current parent.[29,30] This PDE was extended into a variant with self-adaptive crossover and mutation.[31] Multi-objective DE has been used to minimize the error and the number of hidden units in neural network training. The resulting Pareto-optimal front is the trade-off between these two objectives.[32] The nondominated sorting, ranking, and elitism techniques utilized in the NSGA-II have also been incorporated into a DE method.[33] Another approach involving Pareto-based evaluation was applied to an Enterprise Planning problem with the two objectives of cycle time and cost,[34] and compared with the Strength-Pareto Evolutionary Algorithm.[35] Some preliminary work has also reported on the behavior of a nondominated sorting DE multi-objective algorithm,[7] demonstrating its potential worth as a rotationally invariant optimizer for multi-objective problems. For the purposes of this study, we have employed the same approach to integrating DE with NSGA-II.[7] *popsize* individuals are generated from the DE operator and evaluated on the test problem. Sorting and selection occurs using the NSGA-II procedure.

7. Performance Assessment

The set of solutions generated by a multi-objective evolutionary algorithm cannot be evaluated with respect to a single measure of performance. For instance, one algorithm may generate a set, which covers the Pareto-optimal front very well but has poor convergence. In contrast, another algorithm may produce a set with worse coverage, but better convergence toward the Pareto-optimal front. A large number of performance metrics have been proposed in the literature for dealing with this issue and the reader is referred to a recent comparative survey.[36]

The situation for performance assessment is made more difficult as the objective space dimension increases, where a formal proof has demonstrated that for M objectives, at least M performance metrics are required in order to compare two solution sets.[20] Although this is correct, it has also been argued that it is still possible to compare two approximation sets with two performance metrics in order to draw useful conclusions about an algorithm's performance.[19] We have followed this observation in our performance assessment criteria and employed the generational distance metric,[38] which was also used in Ref. 19.

This metric allows us say something useful about the coverage of the approximation set and its convergence to the Pareto-optimal set. The generational distance metric is described in Eq. (5). It measures the average distance of the solutions in the nondominated solution set Q, to a Pareto-optimal solution set P^* (Fig. 14(a)). The solution set P^* is a large sample of uniformly spaced Pareto-optimal solutions

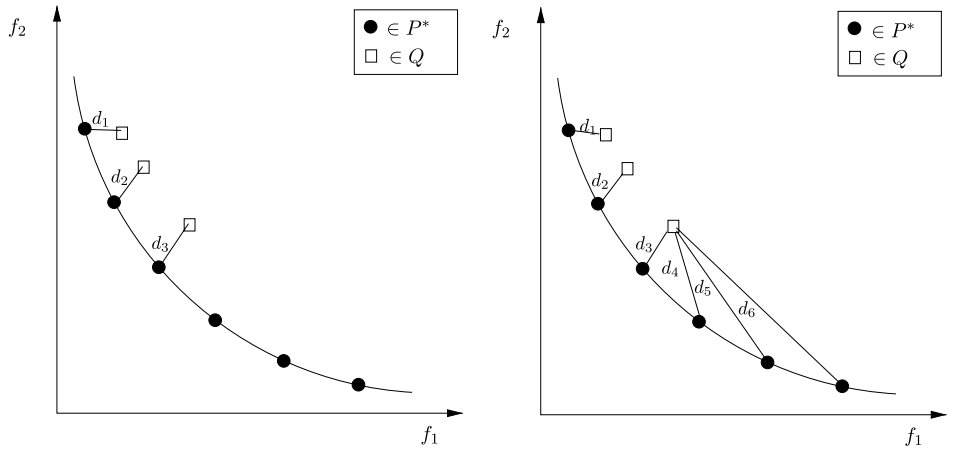

Fig. 14. (a) $GD(Q, P^*)$ measures the minimum distance from each point in Q to a point in P^*, and averages over all minimum distances. This is a measure of convergence of set Q toward set P^*. (b) $GD(P^*, Q)$ measures the minimum distance from each point in P^* to a point in Q, and averages over all minimum distances. This will give a larger value than $GD(Q, P^*)$ because it considers the coverage of the set Q over the set P^*.

in the objective space. $f_m^{*(k)}$ is the mth objective function value of the kth member of the Pareto-optimal solution set, P^*:

$$GD(Q, P^*) = \frac{\sum_{i=1}^{|Q|} d_i}{|Q|}, \tag{5}$$

where

$$d_i = \min_{k=1}^{|P^*|} \sqrt{\sum_{m=1}^{M} (f_m^{(i)} - f_m^{*(k)})^2}.$$

The metric in Eq. (5) is used to measure convergence to the Pareto-optimal solution set. It also has the elegant property that the inverse of this metric in Eq. (6) can be used in order to emphasize a measure of the coverage of a nondominated solution set Q over the Pareto-optimal solution set P^*.[39]

$$GD(P^*, Q) = \frac{\sum_{i=1}^{|P^*|} d_i}{|P^*|}, \tag{6}$$

where

$$d_i = \min_{k=1}^{|Q|} \sqrt{\sum_{m=1}^{M} (f_m^{*(i)} - f_m^{(k)})^2}.$$

The inverse measure is achieved by simply swapping the order in which one measures distances between the two sets. In Eq. (6), we measure the average distance of the solutions in the Pareto-optimal solution set P^*, to a nondominated solution solution set Q. $f_m^{(k)}$ is the mth objective function value of the kth member of the nondominated set, Q. From Fig. 14(a), the $GD(Q, P^*)$ measure results in smaller d_i measures. In other words, the focus of the metric is to measure convergence because only the closest Pareto-optimal solution to a nondominated solution is chosen for d_i. It is apparent from Fig. 14(b) that the $GD(P^*, Q)$ measure assigns the closest nondominated solution from Q, to a Pareto-optimal solution in P^* for the d_i measure. As can be seen from the figure, this inverse measure places emphasis on the poor coverage of set Q on set P^*. If both measures approach 0, the implication is that the nondominated solution set has both a good coverage and convergence to the Pareto-optimal front.

8. Experiments

A population size of 100 individuals was used for each of the algorithms on each of the test problems. A number of the crossover techniques investigated here have not previously been studied within the NSGA-II framework, and we expect that some of the choices of our parameter settings might be sub-optimal for the problems explored. It is not our intention to perform a comparative study in order to find the best parameter settings of these crossover techniques, but we do expect nonspecifically tuned settings to demonstrate improvements in the performance of the NSGA-II with the rotationally invariant behavior of the DE, SPX, UNDX-m, and PCX operators. A number of appropriate parameter settings have been reported for these operators and we have utilized these reported settings where possible. We leave a more detailed comparative study of these operators on rotated multi-objective problems, as an area of future study.

For the DE variant of NSGA-II, 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.[6] In the SPX operator, the simplex size, ϵ, determines the size of the expanded simplex, and we have used $\sqrt{N+1}$, which was used in previous studies in the single-objective domain. A mutation rate of $1/N$ was also used with the NSGA-II incorporating the new SBX with uniform crossover, and the SPX, UNDX-m, and PCX variants. SBX employed a crossover rate of 0.8. For the UNDX-m operator, the parameters $\sigma_\xi = \frac{1}{\sqrt{m}}$ and $\sigma_\eta = \frac{0.35}{\sqrt{N-m}}$ were recommended in Ref. 17 and we have used these values in this study. For the PCX, the σ_ξ and σ_η parameters were set to 0.7 and 0.2, respectively. The PCX variant is sensitive to the σ_ξ parameter. If σ_ξ is too small the offspring generated do not spread across the Pareto-optimal front. In the UNDX-m and PCX version of NSGA-II, m was assigned a value of 3.

Experiments were conducted on the unimodal function described in Sec. 3, the discontinuous and nonuniform mapped problems described in Sec. 4, and the

scalable problem described in Sec. 5. Each of these problems incorporates a multi-objective valley, which is designed to trap points from progressing along the Pareto-optimal front. The bi-objective problems are 10-dimensional in the decision space. The scalable problem with four objectives has eight decision space parameters, and the eight-objective version has 12 decision space variables. Rotations for each of these test problems were performed in the decision space, on each plane, using a random uniform rotation matrix generated using the technique described in Sec. 4, which introduces parameter interactions between all parameters. In N-dimensions, there are $(N-1)N/2$ planes of rotation, introducing parameter interactions between each parameter with every other. Each algorithm was run 50 times on each test bi-objective problem, for a total of 800 generations (80,000 problem evaluations) for each run. The scalable test problems with more than two objectives were run for 300 generations (30,000 problem evaluations). A new random uniform rotation matrix was generated for each run of each algorithm.

9. Discussion and Results

9.1. *Problem (1)*

In Figs. 15 and 16, two parents are located on the Pareto-optimal front, and offspring are generated around these parents. In the nonrotated case of Fig. 15, the NSGA-II with SBX crossover operation generates only nondominated solutions, which are located on the Pareto-optimal front. Unfortunately, when the problem is rotated (Fig. 16), many of the solutions generated are far from the Pareto-optimal front.

Fig. 17, it is apparent that in the presence of a multi-objective ridge, nondominated solutions are generated away from the Pareto-optimal front. This clearly demonstrates the degraded performance of the NSGA-II with SBX crossover according to the analysis presented in Sec. 3. A representative run of the NSGA-II with SBX on Problem (1) also demonstrates this effect in Fig. 18.

Furthermore, in Fig. 18, each of the rotationally invariant crossover operators yields superior performance over the SBX on this problem, with respect to both convergence to the Pareto-optimal front and coverage across the front. Figure 19 also demonstrates that the variation in both the convergence and the coverage is relatively low on each of the rotationally invariant crossover operators.

9.2. *Problem (2)*

The discontinuous problem introduces similar difficulties to the NSGA-II, as the unimodal problem. When the front is discontinuous, there is a tendency for NSGA-II with SBX to generate degraded nondominated solutions, in a similar manner to the search for nondominated solutions on Problem (1). This behavior is exacerbated because the discontinuities do not allow a collapse of the degraded region toward a more ideal solution, thereby enabling the degraded region to be extended further.

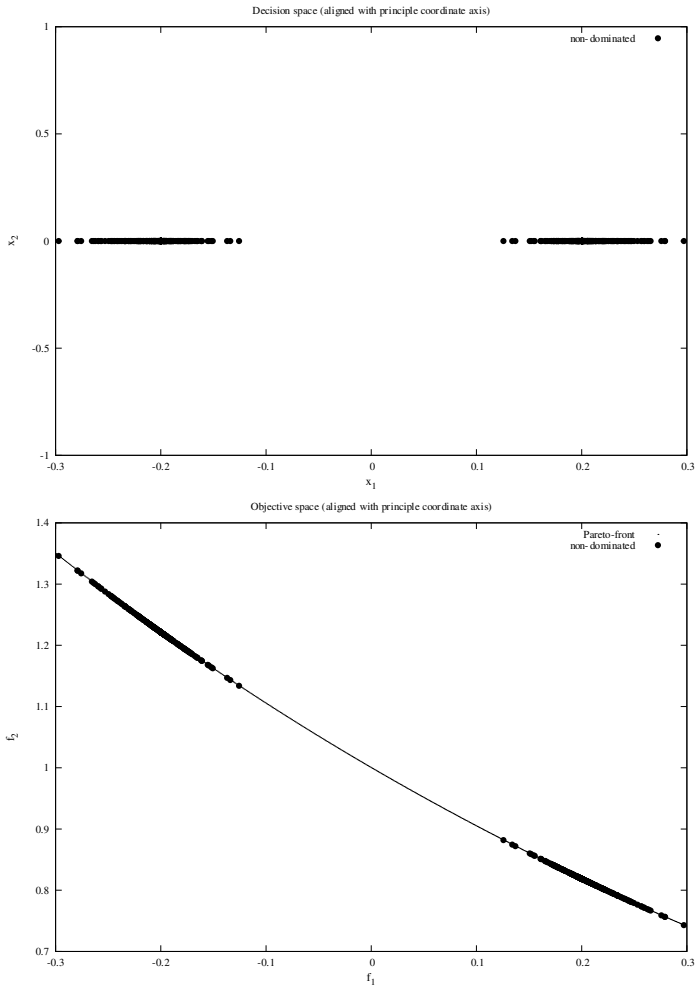

Fig. 15. Parents are located at [−0.2, 0] and [0.2, 0] in the nonrotated case. In the nonrotated case, all solutions generated using SBX by the parents are nondominated and Pareto-optimal.

We can see the effect of this in a representative run of NSGA-II with SBX in Fig. 20. In Fig. 21, it is apparent that there is a high variation in the convergence of the runs of NSGA-II with SBX on this problem, and convergence is generally worse than the other variants. NSGA-II with SBX fails to cover the left most front in many cases (Fig. 20). It typically misses the left most front because of the degradation in performance caused by generating many nondominated solutions away from the Pareto-optimal front (Sec. 3).

From Fig. 21, it is apparent that the variant incorporating DE has the lowest variation in both convergence to the Pareto-optimal fronts and coverage of the Pareto-optimal fronts, as well as the best coverage and convergence overall.

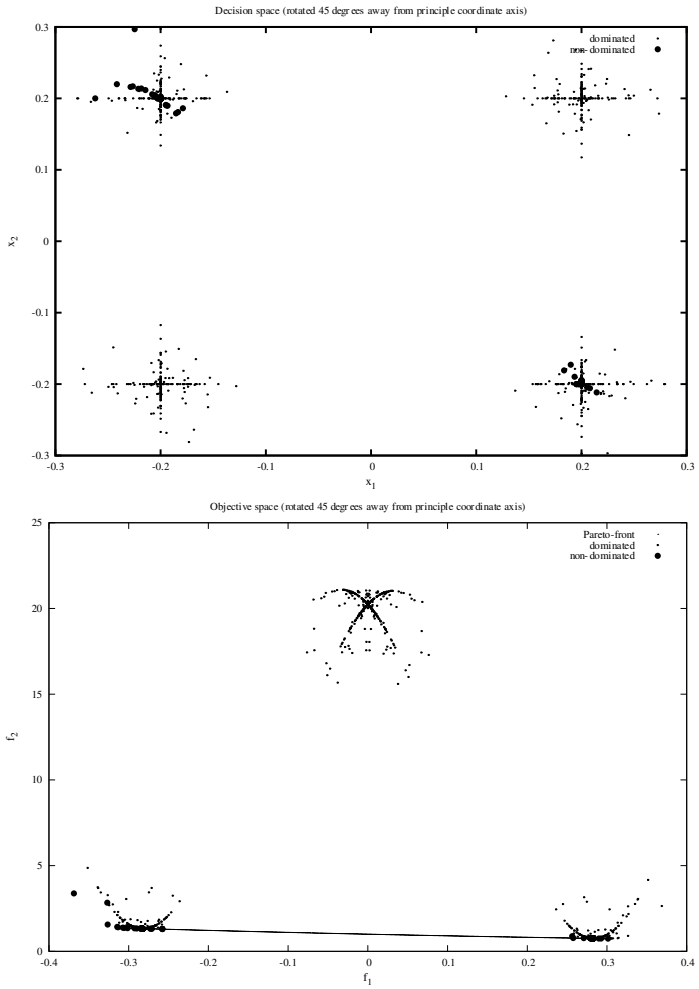

Fig. 16. Parents are located on the Pareto-optimal front in the decision space in the rotated-case at $[-0.2, 0.2]$ and $[0.2, -0.2]$. In the rotated case, far fewer nondominated solutions are generated.

9.3. *Problem (3)*

The results from Figs. 22 and 23 suggest that rotation does not introduce significant difficulties for the NSGA-II with SBX on Problem (3) with respect to convergence toward the Pareto-optimal front. SPX, UNDX-m, and PCX seem to struggle with respect to finding a good coverage of the Pareto-optimal front. The NSGA-II with SBX actually has a comparable variation in the coverage of the fronts with the PCX variant and has a larger variation in coverage with respect to the UNDX and SPX variants. In this problem, the density of solutions toward higher f_1 values is greater and toward lower f_1 values it becomes less. If one considers how offspring are actually generated by SPX, UNDX-m, and PCX, it becomes clearer why these

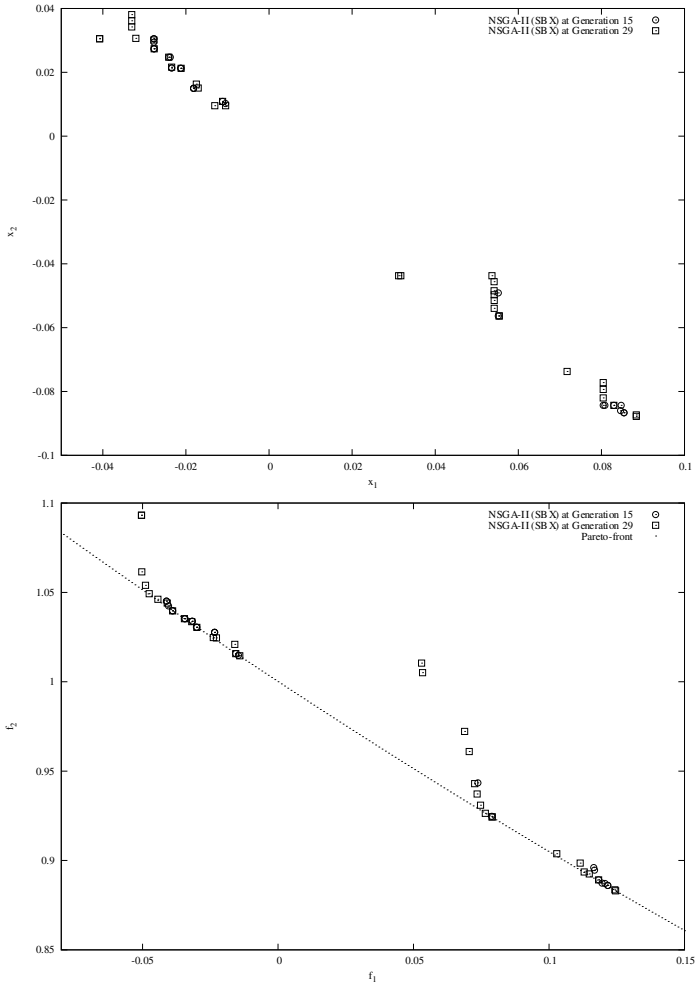

Fig. 17. NSGA-II finds solutions that are nondominated but skew away from the Pareto-optimal front on Problem (1). This figure demonstrates the degradation in performance of NSGA-II over successive generations when a ridge is encountered in a multi-objective problem.

operators struggle to produce offspring toward the lower f_1 region. In the situation where the majority of parents come from the Pareto-optimal front, there is a smaller likelihood of selecting parents, which are in the low density region. Secondly, offspring have a small chance of being generated at the extremes using the centroid-centric approaches.

From Fig. 22, one can see that the DE variant performed well on this problem, with a low variation for convergence to the Pareto-optimal front, and a low variation for coverage of the Pareto-optimal front, with the majority of runs typically covering the front better than the other variants. Because DE uses vector addition, and the

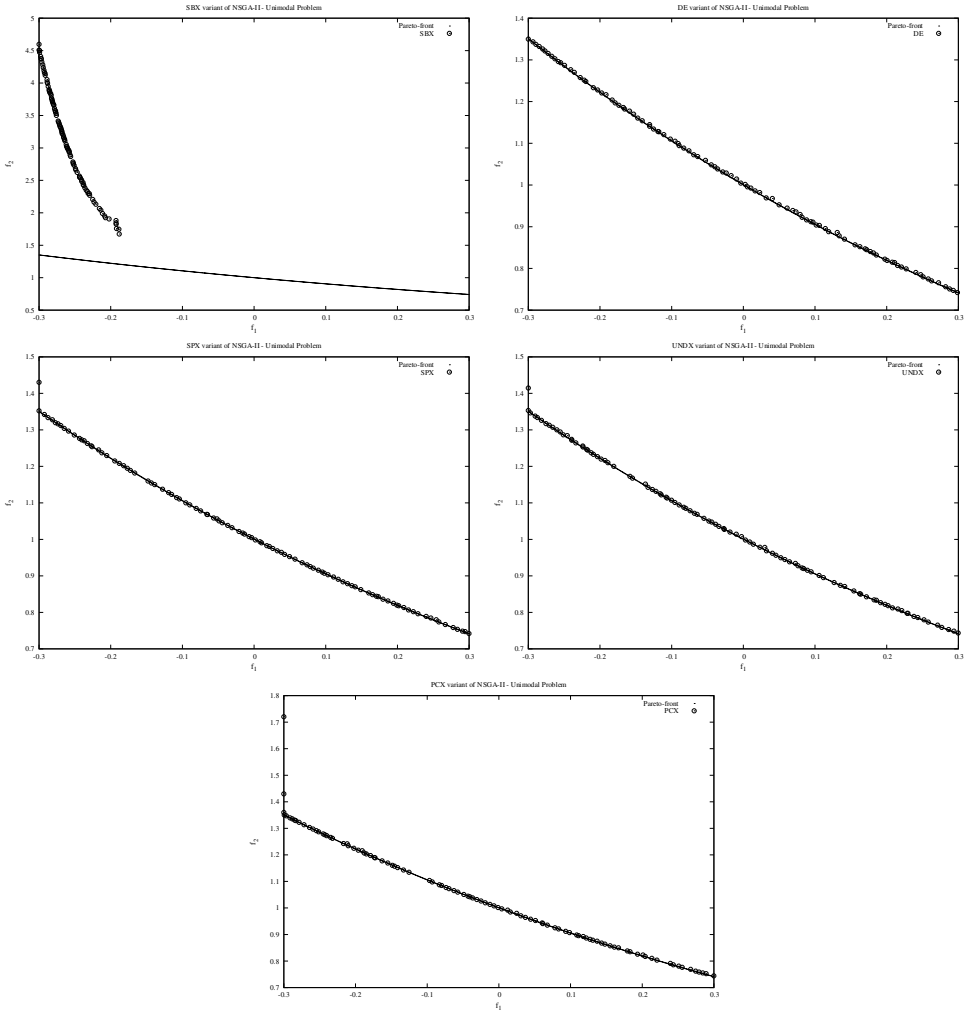

Fig. 18. Nondominated solutions found on Problem (1) after 800 generations.

front is situated on a hyperplane through the decision space, vector differential on this line can easily generate solutions that cover the low density region as well.

9.4. *Problem (4)*

Figure 24 describes the result on Problem (4) with four-objectives when the problem is unrotated. It is apparent that SBX results in the best performance on this problem with respect to the finding a good convergence toward and coverage the Pareto-optimal front using the $GD(Q, P^*)$ and $GD(P^*, Q)$ metrics. Furthermore,

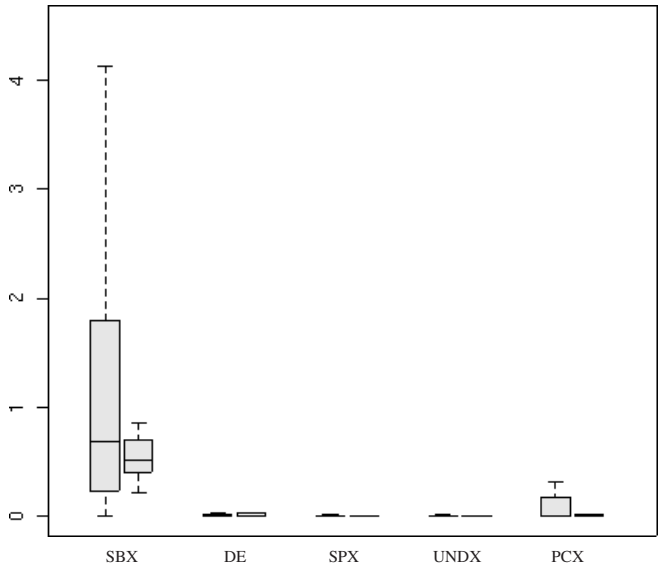

Fig. 19. Boxplots of the $GD(Q, P^*)$ and $GD(P^*, Q)$ measures of the 50 runs of each algorithm variant on Problem (1). The boxplot of the $GD(Q, P^*)$ measures are on the left and the $GD(P^*, Q)$ measures are on the right for each labeled variant.

the variation in both these measures is low on the SBX according to the boxplots. This result is to be expected, as a problem aligned with the principle coordinate axes favors any reproduction operator which generates the majority of its offspring along the principle coordinate axes. When the four-objective problem is rotated the result is quite different, as is apparent from Fig. 25. The performance of the DE operator demonstrates it is now competitive with SBX with respect to both performance metrics. Furthermore, the variation in both these measures increases for the SBX operator. This demonstrates that SBX has difficulties dealing with the introduction of the valley associated with the hyper-ellipsoid, as well as the various optimal step sizes required in each plane.

In the eight-objective problem that has not been rotated, it is apparent that SBX and DE have comparable performance from Fig. 26, with respect to both measures. Interestingly, the $GD(P^*, Q)$, which measures the coverage of the approximation set, produces a similar result for the SPX, UNDX, and PCX operators. This indicates that each of these operators can produce nondominated solutions with similar effectiveness. Figure 27 displays the boxplots for the $GD(Q, P^*)$ and $GD(P^*, Q)$ on the rotated eight-objective problem. It is apparent from this figure that the performance of SBX and DE becomes comparable with SPX, UNDX, and PCX. It cannot be conclusively stated that any of these operators are better than others from this result.

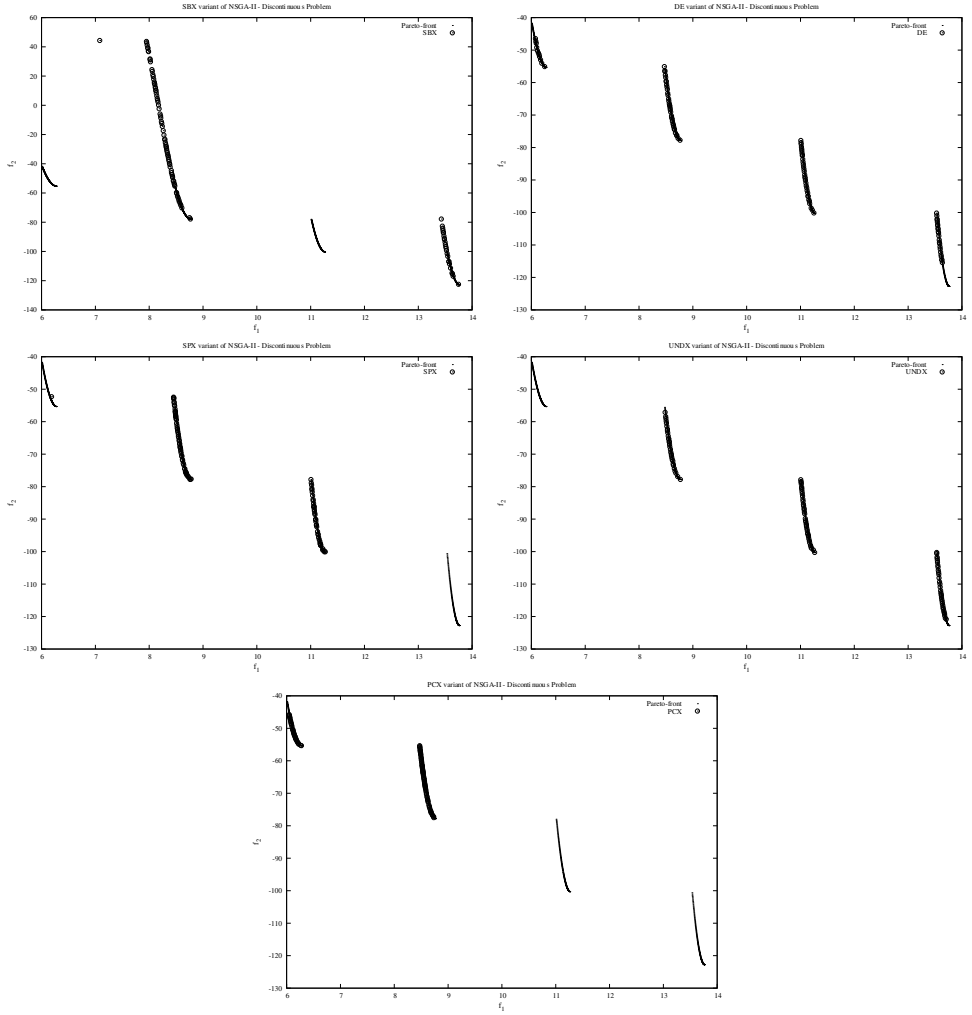

Fig. 20. Nondominated solutions found on Problem (2) after 800 generations.

10. Conclusion

This paper has described an empirical study of the effects that rotation of problems has on the NSGA-II with SBX, in the presence of valleys which can trap the search, on three problems with the properties of unimodality, discontinuous Pareto-optimal fronts, and a nonuniform mapping between the objective and decision space. We have demonstrated that on these three problems that the UNDX-m, SPX, and DE were relatively as good or better than the SBX, taking into consideration the performance metrics for convergence to the Pareto-optimal front and distribution of solutions across the front.

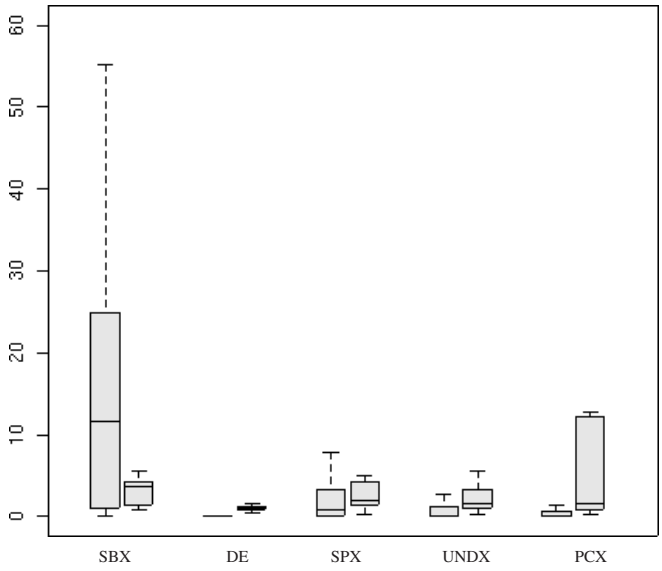

Fig. 21. Boxplots of the $GD(Q, P^*)$ and $GD(P^*, Q)$ measures of the 50 runs of each algorithm variant on Problem (2). The boxplot of the $GD(Q, P^*)$ measures are on the left and the $GD(P^*, Q)$ measures are on the right for each labeled variant.

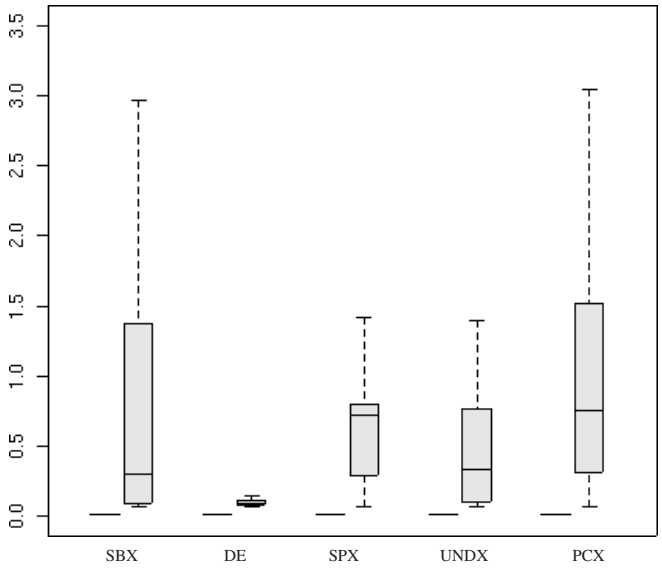

Fig. 22. Boxplots of the $GD(Q, P^*)$ and $GD(P^*, Q)$ measures of the 50 runs of each algorithm variant on Problem (3). The boxplot of the $GD(Q, P^*)$ measures are on the left and the $GD(P^*, Q)$ measures are on the right for each labeled variant.

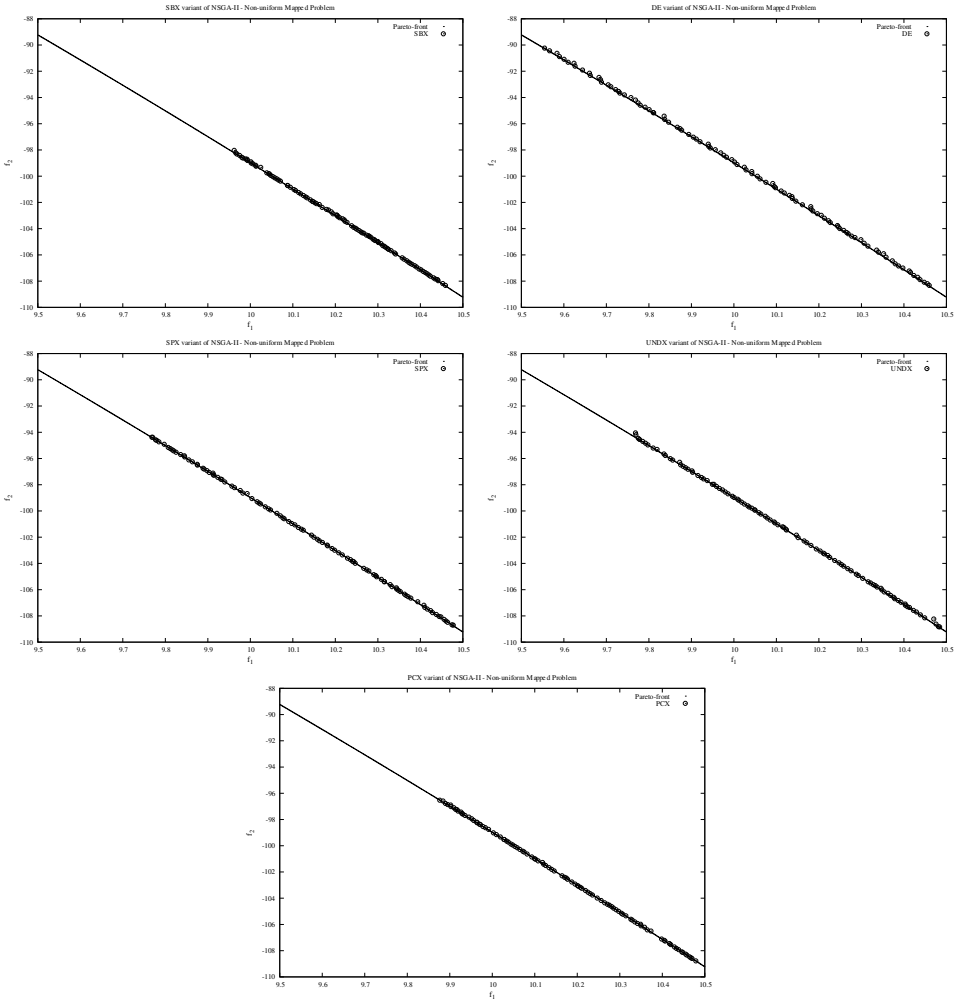

Fig. 23. Nondominated solutions found on Problem (3) after 800 generations.

Multi-objective optimization problems, which are not aligned with the principal coordinate axes, introduce some new issues to multi-objective optimization algorithms, which do not use rotationally invariant reproduction operators. Although it may be possible to discover nondominated solutions using a nonrotationally invariant approach, the nondominated solutions that are generated are less likely to be Pareto-optimal. We have discussed one type of problem exhibiting such behavior in Sec. 3, but one could also consider many other kinds, which will exhibit different characteristics, and a study of such problems may be worthwhile to practitioners working with real-world multi-objective problems, which are typically not aligned with any coordinate system.

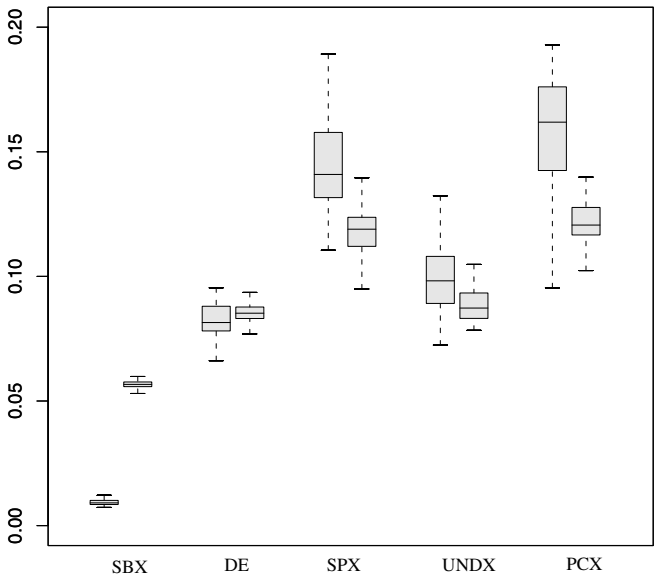

Fig. 24. Boxplots of the $GD(Q, P^*)$ and $GD(P^*, Q)$ measures of the 50 runs of each algorithm variant on unrotated Problem (4) with four objectives. The boxplot of the $GD(Q, P^*)$ measures are on the left and the $GD(P^*, Q)$ measures are on the right for each labeled variant.

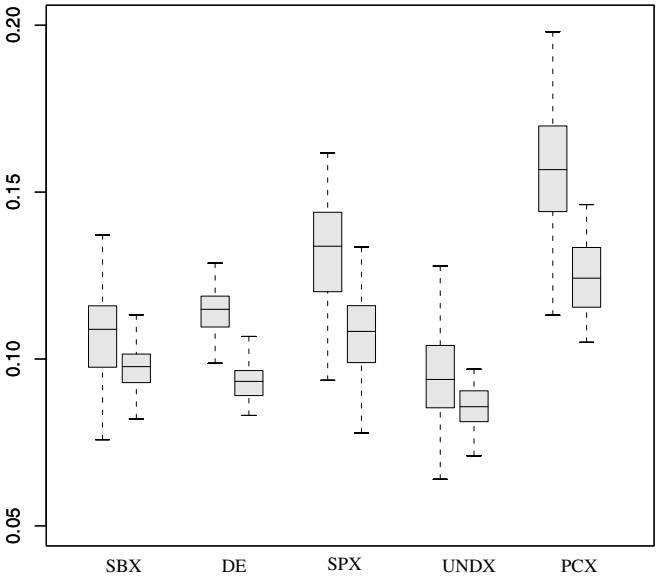

Fig. 25. Boxplots of the $GD(Q, P^*)$ and $GD(P^*, Q)$ measures of the 50 runs of each algorithm variant on rotated Problem (4) with four objectives. The boxplot of the $GD(Q, P^*)$ measures are on the left and the $GD(P^*, Q)$ measures are on the right for each labeled variant.

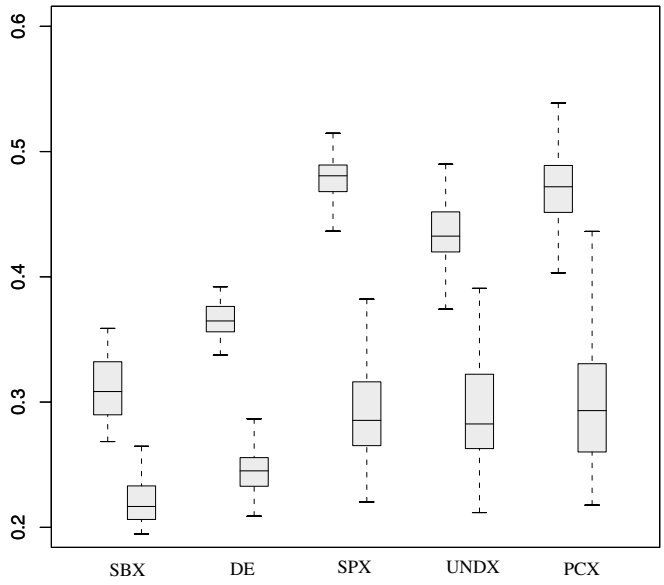

Fig. 26. Boxplots of the $GD(Q, P^*)$ and $GD(P^*, Q)$ measures of the 50 runs of each algorithm variant on unrotated Problem (4) with eight objectives. The boxplot of the $GD(Q, P^*)$ measures are on the left and the $GD(P^*, Q)$ measures are on the right for each labeled variant.

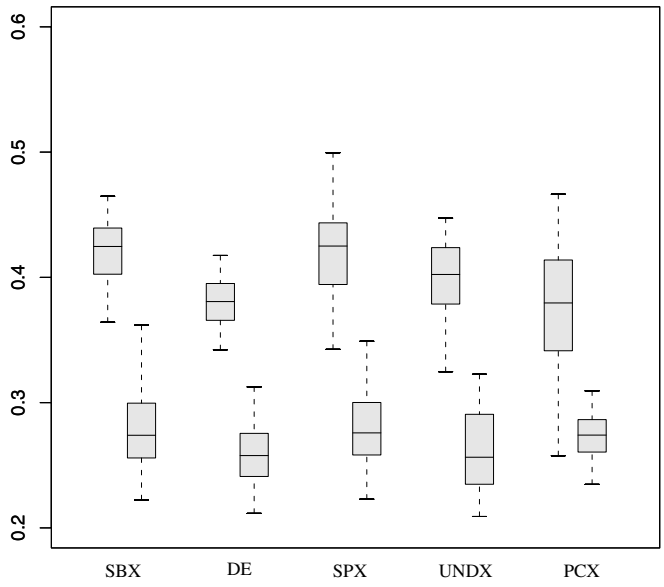

Fig. 27. Boxplots of the $GD(Q, P^*)$ and $GD(P^*, Q)$ measures of the 50 runs of each algorithm variant on rotated Problem (4) with eight objectives. The boxplot of the $GD(Q, P^*)$ measures are on the left and the $GD(P^*, Q)$ measures are on the right for each labeled variant.

The PCX variant seemed not to perform as well as the SPX, UNDX-m, and DE variants on the bi-objective problems. This is understandable when one considers the distribution of children generated from parents located on the Pareto-optimal front. Once the majority of potential parents have converged toward the Pareto-optimal front, the control parameters, which determine the extent of the distribution of children, are no longer suitable for efficiently generating solutions along the Pareto-optimal front on the bi-objective problems investigated in this study. This is also true for the SPX and UNDX-m variants but less so because these operators generate children more efficiently along the Pareto-optimal front on the bi-objective problems because of the elongated shape of the distribution of children generated around parents located on the Pareto-optimal fronts of these problems.

The nature of the Pareto-optimal fronts of the described bi-objective test problems may have favored the performance of DE. The vector-wise property of DE will generate new children on the Pareto-optimal front easily, because the Pareto-optimal set is piece-wise linear in the decision space. In the case of UNDX-m, SPX, and PCX, offspring generated from parents (that are located on the Pareto-optimal front) are less likely to be located on the Pareto-optimal front. This consequently explains the good performance of Differential Evolution even over 50 runs, which consistently outperformed the other recombination operators on each of the three bi-objective problems, providing a low variation in both the convergence and coverage of the Pareto-optimal fronts, as well as relatively high quality solution sets for both of these features. Obviously, a future avenue of work would be introducing nonlinearities to the Pareto-optimal set, as suggested in Ref. 40.

On the scalable test Problem (4), it is apparent that in four objectives SBX is the best performer on the unrotated problem, but the advantage associated with SBX disappears when the problem is rotated. As the objective space dimension is increased to eight objectives the difference in performance between the variants becomes negligible with respect to the performance metrics and whether the problem is rotated or unrotated.

As the objective space dimension increases it is obvious that a rotationally invariant recombination operator will have little impact on the performance of EMO algorithms incorporating nondominated sorting, as such algorithms tend to find an exponentially increasing number of nondominated solutions. This results in an inability of the algorithm to differentiate between solutions. It is possible that alternative selection schemes that are not based on domination principles might be more effective when used with the recombination operators suggested in this study for problems with large numbers of objectives.

Acknowledgment

The authors would like to thank Hajime Kita for providing the source code for the UNDX-m.

References

1. R. Salomon, Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions: A survey of some theoretical and practical aspects of genetic algorithms, *Bio. Systems* **39**(3) (1996) 263–278.
2. K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, A fast and elitist multi-objective genetic algorithm: Nsga-ii, *IEEE Trans. Evol. Comput.* **6**(2) (2002) 182–197.
3. K. Deb and R. B. Agrawal, Simulated binary crossover for continuous search space, *Complex Syst.* **9**(2) (1995) 115–148.
4. K. Deb and A. Kumar, Real-coded genetic algorithms with simulated binary crossover: Studies on multi-modal and multi-objective problems, *Complex Syst.* **9**(6) (1995) 431–454.
5. F. R. Deutsch, *Best Approximation in Inner Product Spaces* (Springer, 2001), p. 51.
6. D. Corne, M. Dorigo and F. Glover (eds.), *New Ideas in Optimization* (McGraw-Hill, London, UK, 1999), p. 98.
7. A. Iorio and X. Li, Solving rotated multi-objective optimization problems using differential evolution, in *AI 2004: Advances in Artificial Intelligence: 17th Australian Joint Conf. Artificial Intelligence* (Cairns, Australia, Heidelberg, 2004), p. 861.
8. A. Iorio and X. Li, Rotated test problems for assessing performance of multi-objective optimization algorithms, in *Proc. 2006 Genetic and Evolutionary Computation Conf. (GECCO'06)*, pp. 683–690.
9. A. Iorio and X. Li, Rotationally invariant crossover operators in evolutionary multi-objective optimization, in *Proc. 6th Int. Conf. on Simulated Evolution and Learning (SEAL'06)*, pp. 310–317.
10. A. Iorio and X. Li, Incorporating directional information within a differential evolution algorithm for multi-objective optimization, in *Proc. 2006 Genetic and Evolutionary Computation Conf. (GECCO'06)*, pp. 691–698.
11. K. Deb and M. Goyal, A combined genetic adaptive search (geneas) for engineering design, *Comput. Sci. Inform.* **26**(4) (1995) 30–45.
12. P. Ballester and J. N. Carter, Real-parameter genetic algorithms for finding multiple optimal solutions in multi-modal optimization, in *Proc. 2003 Genetic and Evolutionary Computation Conf. (GECCO-03)* (2003), pp. 707–717.
13. S. Tsutsui and M. Yamamura, Multi-parent recombination with simplex crossover in real coded genetic algorithms, in *Proc. 1999 Genetic and Evolutionary Computation Conf. (GECCO-99)* (1999), pp. 657–664.
14. I. Ono, H. Kita and S. Kobayashi, *A Real-coded Genetic Algorithm using the Unimodal Normal Distribution Crossover*, Advances in Evolutionary Computing: Theory and Applications (Springer, 2003), pp. 213–237.
15. H. Kita, A comparison study of self-adaptation in evolution strategies and real-coded genetic algorithms, *Evol. Comput.* **9**(2) (2001) 223–241.
16. I. Ono, S. Kobayashi and K. Yoshida, Optimal lens design by real-coded genetic algorithms using undx, *Comput. Meth. Appl. Mech. Eng.* **186** (2000) 483–497.
17. H. Kita, I. Ono and S. Kobayashi, Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms, in *Proc. 1999 Congress on Evolutionary Computation (CEC'99)* (1999), pp. 1581–1587.
18. K. Deb, D. Joshi and A. Anand, Real-coded evolutionary algorithms with parent-centric recombination, Tech. Rep. KanGAL Report No. 2001003, Indian Institute of Technology, Kanpur (2001).

19. K. Deb and S. Jain, Running performance metrics for evolutionary multi-objective optimization, Tech. Rep. KanGAL Report No. 2002004, Indian Institute of Technology, Kanpur (2004).
20. E. Zitzler, M. Laumanns, L. Thiele, C. M. Fonseca and V. G. Fonseca, Why quality assessment of multiobjective optimizers is difficult, in *Proc. 2002 Genetic and Evolutionary Computation Conf. (GECCO'02)* (2002), pp. 666–674.
21. K. Deb, L. Thiele, M. Laumanns and E. Zitzler, Scalable multi-objective optimization test problems, in *Proc. 2002 Congress on Evolutionary Computation CEC2002* (IEEE Press, 2002), pp. 825–830.
22. K. Deb, L. Thiele, M. Laumanns and E. Zitzler, Real-coded evolutionary algorithms with parent-centric recombination, in *Proc. 2002 Congress on Evolutionary Computation CEC2002* (IEEE Press, 2002), pp. 61–66.
23. K. Praditwong and X. Yao, How well do multi-objective evolutionary algorithms scale to large problems, in *Proc. 2007 Congress on Evolutionary Computation CEC2007* (IEEE Press, 2007), to be published.
24. K. V. Price, Differential evolution: A fast and simple numerical optimizer, in *Biennial Conf. North American Fuzzy Information Processing Society*, Vol. 3339 (New York, 1996), pp. 524–527.
25. J. Ilonen, J.-K. Kamarainen and J. Lampinen, Differential evolution training algorithm for feed-forward neural networks, *Neural Process. Lett.* **7**(1) (2003) 93–105.
26. R. Storn, Differential evolution design of an iir-filter, in *Proc. IEEE Int. Conf. Evolutionary Computation ICEC'96* (New York, 1996), pp. 268–273.
27. T. Rogalsky, R. W. Derksen and S. Kocabiyik, Differential evolution in aerodynamic optimization, in *Proc. 46th Annual Conf. Canadian Aeronautics and Space Institute* (1999), pp. 29–36.
28. C. S. Chang and D. Y. Xu, Differential evolution of fuzzy automatic train operation for mass rapid transit system, in *IEEE Proc. Electric Power Appl.* **147**(3) (2000) 206–212.
29. H. A. Abbass, R. Sarker and C. Newton, Pde: A pareto-frontier differential evolution approach for multi-objective optimization problems, in *Proc. 2001 Congress on Evolutionary Computation (CEC'2001)*, Vol. 2 (2001), pp. 971–978.
30. H. A. Abbass and R. Sarker, The pareto differential evolution algorithm, *Int. J. Artif. Intell. Tools* **11**(4) (2002) 531–552.
31. H. A. Abbass, The self-adaptive pareto differential evolution algorithm, in *Proc. 2002 Congress on Evolutionary Computation (CEC'2002)*, Vol. 1 (2002), pp. 831–836.
32. H. A. Abbass, A memetic pareto evolutionary approach to artificial neural networks, in *Proc. Australian Joint Conf. Artificial Intelligence* (Adelaide, Australia), Lecture Notes in Artificial Intelligence, Vol. 2256 (Springer-Verlag, 2001), pp. 1–12.
33. N. K. Madavan, Multi-objective optimization using a pareto differential evolution approach, in *Proc. 2002 Congress on Evolutionary Computation (CEC'2002)*, Vol. 2 (IEEE Press, 2002), pp. 1145–1150.
34. F. Xue, Multi-objective differential evolution and its application to enterprise planning, in *Proc. 2003 IEEE Int. Conf. Robotics and Automation (ICRA'03)*, Vol. 3 (IEEE Press, 2003), pp. 3535–3541.
35. F. Xue, A. C. Sanderson and R. J. Graves, Pareto-based multi-objective differential evolution, in *Proc. 2003 Congress on Evolutionary Computation (CEC'2003)*, Vol. 2 (IEEE Press, 2003), pp. 862–869.
36. E. Zitzler, L. Thiele, M. M. Laumanns, C. M. Fonseca and V. G. Fonseca, Performance assessment of multi-objective optimizers: An analysis and review, *IEEE Trans. Evol. Comput.* **2**(2) (2003) 117–132.

37. X. Yao and Y. Liu, Fast evolution strategies, in *Evolutionary Programming VI*, Lecture Notes in Computer Science, Vol. 1213 (Springer, Berlin, 1997), pp. 151–161.
38. D. V. Veldhuizen, Multi-objective evolutionary algorithms: Classifications, analyses, and new innovations, Ph.D. Dissertation, Airforce Institute of Technology (1999).
39. X. Li and J. Branke, Performance metrics and particle swarm methods for dynamic multi-objective optimization problems, Tech. Rep. no. TR-05-6, RMIT University, Melbourne Australia (2005).
40. K. Deb, A. Sinha and S. Kukkonen, Multi-objective test problems, linkages, and evolutionary methodologies, in *Proc. 2006 Genetic and Evolutionary Computation Conf. (GECCO'06)*, pp. 1141–1148.
41. K. Deb, S. Chaudhuri and K. Miettinen, Towards estimating nadir objective vector using evolutionary approaches, in *Proc. 2006 Genetic and Evolutionary Computation Conf. (GECCO'06)*, pp. 643–650.
42. K. Deb, Multi-objective genetic algorithms: Problem difficulties and construction of test problems, *Evol. Comput.* **7**(3) (1999) 205–230.