

# Improving the performance and scalability of Differential Evolution on problems exhibiting parameter interactions

Antony W. Iorio · Xiaodong Li

Published online: 17 June 2010  
© Springer-Verlag 2010

**Abstract** Differential Evolution (DE) is a powerful optimization procedure that self-adapts to the search space, although DE lacks diversity and sufficient bias in the mutation step to make efficient progress on non-separable problems. We present an enhancement to DE that introduces greater diversity while also directing the search to more promising regions. The Combinatorial Sampling Differential Evolution (CSDE) is introduced which can sample vectors in two ways; highly correlated with the search space or around a ‘better’ individual. The CSDE approach can provide a similar number of samples as crossover, without being biased towards the principle coordinate axes of a decision space. This approach to sampling vectors is capable of optimizing problems with extensive parameter interactions. It also demonstrates fast convergence towards the global optimum and is highly scalable in the decision space on a variety of single and multi-objective problems due to the balance between sampling highly directed correlated vectors and non-correlated vectors which contribute to sampling diversity.

**Keywords** Evolutionary Algorithm · Differential Evolution · Optimization · Non-separable problem · Parameter interactions

---

A. W. Iorio (✉)  
Defense and Security Applications Research Centre, University of New South Wales @ Australian Defense Force Academy, Northcott Drive, Canberra, ACT 2600, Australia  
e-mail: a.iorio@adfa.edu.au

X. Li  
School of Computer Science and Information Technology, RMIT University, GPO Box 2476v, Melbourne, VIC 3001, Australia

## 1 Introduction

Despite the power of many population-based stochastic optimization algorithms, they can meet with difficulties on optimization problems which are non-separable. Traditional Genetic Algorithms fail to optimize these problems efficiently because they typically perform independent perturbations of decision variables. Unfortunately, many real-world problems are not linearly separable. On problems which are not aligned with the principle coordinate axes, the small mutation rates frequently used in Genetic Algorithms are known to be even less efficient than a random search (Salomon 1996). One approach for optimizing such problems is to use a vector-based scheme such as Differential Evolution (DE).

Despite the fact that DE has many attractive characteristics, it also has a number of limitations which we will outline here and attempt to address in this paper.

First, crossover is needed to introduce sufficient diversity. Unfortunately, crossover is not a rotationally invariant operation.

The use of crossover in DE introduces diversity to the population, far more than mutation alone. As the decision space dimension scales, the importance of having a diverse population from which to sample becomes significantly important to make efficient progress towards more optimal solutions in the search space. Unfortunately, because the offspring that crossover can generate are dependent on the principle coordinate axes, crossover provides little benefit to the optimization of non-separable problems.

For an algorithm to be rotationally invariant in the context of optimization algorithms, it should produce offspring in the same relative location, irrespective of the orientation of the fitness landscape. In addition, requiring

that a DE algorithm be strictly rotationally invariant introduces further problems.

Although rotationally invariant DE approaches provide vector-wise samples which are not biased with respect to any particular coordinate axes they also lowers the number of potential offspring dramatically because they do not use crossover (Lampinen and Zelinka 2000). This leads to the second limitation associated with DE; a mutation-only strategy is insufficient to make progress because it does not provide sufficient sampling diversity to explore the search space and can lead to stagnation.

Our contention is that for an optimization algorithm to perform efficiently on a non-separable problem, it must not exhibit the extreme dependency on the principle coordinate axes that is typically the case with crossover. In addition, we contend that it is unnecessary for it to be strictly rotationally invariant, as long as it is capable of generating sufficient diversity.

Clearly, these two limitations of DE are diametrically opposed, which gives us an indication of the type of algorithm that can address them. Such an algorithm must maintain a balance between both requirements.

Ideally, we would like a scheme which is biased to accelerate convergence, is capable of generating a diverse variety of offspring solutions in a manner which minimizes distribution bias, and is capable of optimizing non-separable and separable problems equally well. Furthermore, it should be simple to implement and computationally efficient and capable of dealing with multi-objective search spaces. Typically, in real-world problem solving domains, multi-objective problems are the norm rather than the exception, which compounds the difficulty of many problems that also exhibit parameter interactions. Our final requirement is that DE should also utilise the unique aspects of multi-objective search spaces to find non-dominated solution sets efficiently.

As we mentioned earlier, traditional crossover offers sampling diversity, but is really only effective on separable problems because of the way it generates points. It would be desirable for DE to have the capability of producing a large number of samples, while still remaining effective on non-separable problems in high decision space dimensions.

### 1.1 Outline

To begin with, DE is introduced in Sect. 2, as well as the issues that can hamper the performance of DE. Following this, clarification of the terminology used in this paper is provided in Sect. 3. In Sect. 4, we describe the issues associated with algorithms that are not rotationally invariant and go on to introduce the Combinatorial Sampling Differential Evolution algorithm, CSDE (Li and Iorio

2008), which attempts to address some of the deficiencies of existing DE approaches. In Sect. 5, the experimental methodology and problems are introduced, including methods employed for evaluating algorithm performance. Sections 6, 7 and 8 provide the comparative analysis and results of our experiments, followed by the important implications and conclusions outlined in Sect. 9

## 2 Differential Evolution and its limitations

Differential Evolution is a population-based vector-wise algorithm for global optimization as described by Price (1996). It has demonstrated its robustness and power in a variety of applications, such as neural network learning (Ilonen et al. 2003), IIR-filter design (Storn 1996), and the optimization of aerodynamic shapes (Rogalsky et al. 1999).

The DE optimization algorithm works by generating difference vectors between points in the search space, and using the resulting scaled difference vector to perturb existing points in the population (Price 1996). For comparative purposes, a baseline Differential Evolution algorithm (DE) is employed in this study and described here. This algorithm is similar to the DE/rand/1/bin (Price 1999, p. 79–108), but uses a crossover rate of  $CR = 1.0$ , making it rotationally invariant. This baseline algorithm is described in Algorithm 1. In this approach, the population is iterated over and every individual has a chance to participate in the DE calculation. Two randomly selected individuals,  $\mathbf{x}^{(r1)}$  and  $\mathbf{x}^{(r2)}$  are chosen from the population of size  $popsize$  such that they are not equal to each other or  $\mathbf{x}^{(i)}$ . The DE calculation is performed for all  $N$  parameters and the mutation component,  $F(x_j^{(r1)} - x_j^{(r2)})$  is added to the current individual  $x_j^{(i)}$  to perturb  $\mathbf{x}^{(i)}$ .  $x_j^{(L)}$  and  $x_j^{(U)}$  refer to the lower and upper bounds of parameter  $x_j$ , respectively. A boundary repair mechanism is employed when the offspring vector  $u_j^{(i)}$  is outside the boundary.

Although it is unstated in the algorithm description, the offspring replaces the parent only if it is fitter than the parent. In the multi-objective case, the offspring population and parent population are sorted together, and this is described in more detail in Sect. 4.3.

Differential Evolution has a number of attractive features; difference vectors can be correlated with the search space, it uses only  $\mathcal{O}(popsize)$  processes, it does not need a predefined probability distribution for generating offspring, the objective functions do not need to be differentiable, it can provide multiple solutions from a single run of the algorithm, it is very simple to implement, and is a parallel optimization procedure like many other population based schemes.

**Algorithm 1** Baseline Differential Evolution

```

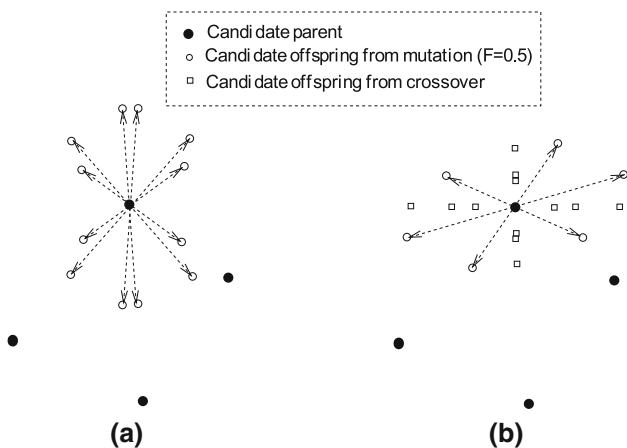
1:  $i = \text{floor}(\text{rand}(0,1] * \text{popsize})$ 
2: for  $k=0$  to  $\text{popsize} - 1$  do
3:   Randomly select  $r1, r2 \in \{1, 2, \dots, \text{popsize}\}$  such that  $r1 \neq r2 \neq i$ 
4:    $j = \text{floor}(\text{rand}(0,1] * N)$ 
5:   for  $n=0$  to  $N - 1$  do
6:     if  $\text{rand}(0,1) < CR$  then
7:        $u_j^{(i)} = x_j^{(i)} + F(x_j^{(r1)} - x_j^{(r2)})$ 
8:     else
9:        $u_j^{(i)} = x_j^{(i)}$ 
10:    end if
11:    if  $u_j^{(i)} < x_j^{(L)}$  then
12:       $u_j^{(i)} = x_j^{(i)} + \text{rand}(0,1) * (x_j^{(L)} - x_j^{(i)})$ 
13:    end if
14:    if  $u_j^{(i)} > x_j^{(U)}$  then
15:       $u_j^{(i)} = x_j^{(i)} + \text{rand}(0,1) * (x_j^{(U)} - x_j^{(i)})$ 
16:    end if
17:     $j = (j + 1) \bmod N$ 
18:  end for
19:   $i = (i + 1) \bmod \text{popsize}$ 
20: end for

```

It is important to elucidate further on one of the reasons for why the rotationally invariant DE/rand/1/bin used in this study performs poorly on non-separable problems. In Fig. 1b (Fig. 1a will be discussed after the CSDE algorithm is introduced), the offspring and parents are represented for a population size of 4. The number of potential unique offspring that can be sampled for a single base-vector by such a scheme is determined by Eq. 1

$$(Np - 1)(Np - 2)(2^D - 1) \tag{1}$$

where  $D$  is the decision space dimension. The term  $2^D$  represents the number of possible offspring that can be generated from binomial crossover. The term  $(Np - 1)(Np - 2)$



**Fig. 1** **a** The offspring generated from a population of 4 using the CSDE scheme. **b** The offspring generated from a population of 4 using the basic DE/rand/1/bin scheme *with* and *without* crossover

is the number of possible offspring that can be generated from vector-wise mutation. In addition, crossover can produce duplicate individuals that were already sampled. In order to not count these individuals, we subtract the duplicates. It deserves to be noted that Eq. 1 is equivalent to previous results which reported upon the number of samples possible in an entire population (Lampinen and Zelinka 2000).

In accordance with this equation, the offspring distributions in Fig. 1b illustrate an enumeration of all 18 possible offspring for a single target vector. The first column of this table details all possible differentials from Fig. 1b. The second contains the location of offspring produced by a mutation operation with  $F = 0.5$ . The third column contains the coordinates of unique offspring resulting from crossover, which do not overlap with offspring resulting from a mutation operation or any existing parents. The total number of offspring possible from a population of four individuals where  $0 < CR < 1.0$  is 18.

From Eq. 1, we can see that as the decision space dimension scales, crossover is responsible for the majority of the offspring individuals that the algorithm can generate through the  $2^D$  term (Lampinen and Zelinka 2000). It is also clear from this figure that crossover samples along the principle coordinate axes, so although it generates many offspring, it also constrains them to this region. It is only capable of independent sampling in each decision space dimension.

If we consider Fig. 1b, where crossover is absent, but rotational invariance is maintained, significantly fewer offspring can be sampled for a single base-vector. The number of potential offspring that can be sampled is equal to  $(Np - 1)(Np - 2)$ . The implication here is that a rotationally invariant DE scheme is highly dependent on the population size to maintain sample diversity. Although it samples offspring independent of any particular coordinate axes, it does not scale in the decision space as well as a scheme incorporating crossover.

### 3 Parameter interaction concepts and non-separability

In the literature, a number of concepts related to parameter interactions have been used, such as non-separability, separability, linkage, and epistasis. Until now, a clear discussion of how they relate to each other has not been provided. To this end, we will define these concepts by showing how they have been used in the literature and discuss how they relate to each other.

We begin this discussion with a definition of additively separable functions and derive a number of theorems which clearly define what additive separability is and the converse of this, namely non-separability.

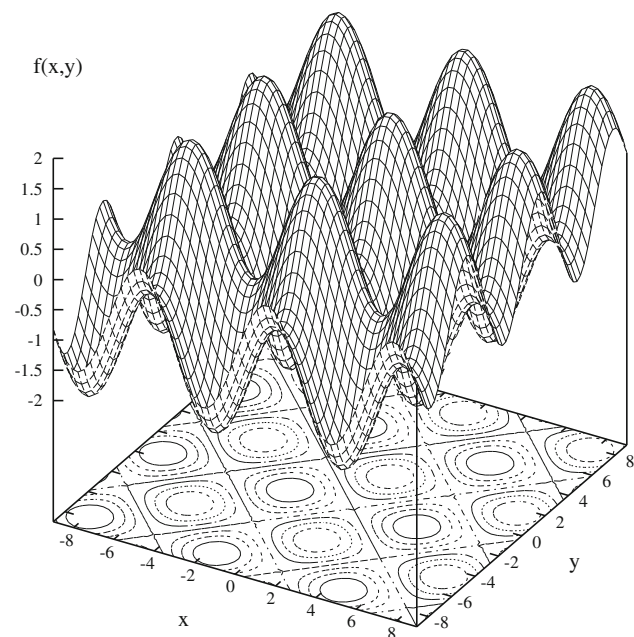
**Definition 2.6.1** (*Additive separable function*) A function of two variables  $F(x, y)$  is called additively separable if it can be represented as  $f(x) + g(y)$  for two single-variable functions  $f(x)$  and  $g(y)$ .

One should note that functions of constants such as  $F(x, y) = 2$  as well as functions of one variable such as  $F(x, y) = h(y)$  are also separable in the additive sense. Furthermore, if  $f(x)$  and  $g(y)$  equal constants in the definition then  $F(x, y)$  is still additively separable. Consider also what separability means geometrically for a function, as detailed in Fig. 2, where for a fixed  $x = a$ , each cross section  $F(a, y)$  is  $g(y) + C$  where  $C$  is some constant translation of the function  $g(y)$  with  $C = f(a)$ . That is, the sections where  $x = a$  for different parameters of  $a$  all result in  $F(a, y)$  exhibiting the same modalities. Clearly, this function is additively separable.

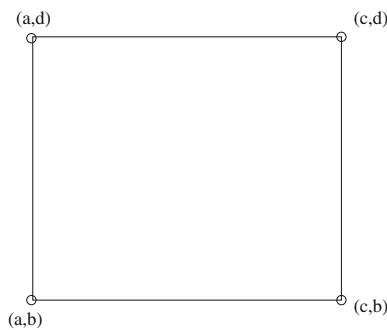
A basic necessary requirement for a function to be additively separable can be obtained if we assume that  $F(x, y) = f(x) + g(y)$  and we then study the behavior of the function at its four extremes  $\{(a, b), (c, b), (c, d), (a, d)\}$  (Fig. 3). From this, we can derive  $F(c, b) + F(a, d) - F(a, b) = (f(c) + g(b)) + (f(a) + g(d)) - (f(a) + g(b)) = f(c) + g(d) = F(c, d)$ . The following theorem follows from this.

**Theorem 2.6.1** *If  $F(x, y)$  is additively separable, then for all  $\{a, b, c, d\}$  it follows that  $F(c, b) + F(a, d) - F(a, b) = F(c, d)$ .*

Using  $a = b = 0$  and  $c = d = 1$  we can see  $F(x, y) = xy$  is non-separable according to our definition, because the



**Fig. 2** Graph of  $\sin(x) + \sin(y)$ , a separable function



**Fig. 3** Four extreme corners  $\{(a, b), (c, b), (c, d), (a, d)\}$  of a function  $F(x, y)$

left hand of this equation equals zero and the right hand side is equal to one. In addition, if we let  $f(x) = (x, 0)$  and  $g(y) = F(0, y) - F(0, 0)$  and use Theorem 2.6.1 with  $x = c, y = d$  and  $a = b = 0$  we get  $F(x, y) = F(x, 0) + F(0, y) - F(0, 0) = f(x) + g(y)$ .

Obviously, if  $F(x, y) = f(x) + g(y)$  then the partial derivatives of  $F$  are easy to compute.  $F_x = f'(x)$  since  $g(y)$  is constant as a function of  $x$ . Similarly,  $F_y = g'(y)$ ,  $F_{xx} = f''(x)$ ,  $F_{yy} = g''(y)$  and more importantly  $F_{xy} = F_{yx} = 0$  when  $F(x, y)$  is separable. This can be re-stated in Theorem 2.6.2.

**Theorem 2.6.2** *If  $F(x, y)$  is additively separable, then  $\frac{d^2F}{dx dy} = 0$ .*

Theorem 2.6.2 provides us with a means to measure the separability of a function and this notion can be expanded upon with the use of a Hessian matrix. Parameter interactions can be measured directly using a Hessian matrix (Newham et al. 2003). The Hessian matrix (Eq. 2) is useful for measuring parameter interactions and is the square matrix of second-order partial derivatives of a function.

$$H(f) = \begin{bmatrix} \frac{d^2f}{dx_1^2} & \frac{d^2f}{dx_1 dx_2} & \cdots & \frac{d^2f}{dx_1 dx_n} \\ \frac{d^2f}{dx_2 dx_1} & \frac{d^2f}{dx_2^2} & \cdots & \frac{d^2f}{dx_2 dx_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{d^2f}{dx_n dx_1} & \frac{d^2f}{dx_n dx_2} & \cdots & \frac{d^2f}{dx_n^2} \end{bmatrix} \tag{2}$$

The mixed second-order derivatives of a function are the entries off the main diagonal in the Hessian, and these derivatives provide a indication of the degree to which parameters are interacting with each other. The second-order derivative test is a criterion often useful for determining whether a given stationary point of a function is a local maximum or a local minimum. If the second-order derivative is less than zero, then the function has a local maximum at that point, if it is greater than zero then it has a local minimum, and if it is zero then the second derivative test says that the variables are separable.

In the last case, the function may have a local maximum or minimum there, but the function is sufficiently ‘flat’ that this is undetected by the second derivative. Such ‘flat’ situations arise where parameters have very little interaction with each other.

In the Hessian matrix, element  $(i, j)$  of  $H$  can be approximated by Eq. 3 where  $f_{ij}$  is the output when parameters with indices  $i$  and  $j$  are perturbed,  $f_i$  and  $f_j$  are the output values when each parameter is perturbed independently, and  $\Delta x_i$  and  $\Delta x_j$  are the parameter perturbations.

$$\frac{d^2f}{dx_i dx_j} \cong \frac{f_{ij} - f_i - f_j + f_o}{\Delta x_i \Delta x_j} \tag{3}$$

From this equation, it is also apparent that if  $f_{ij}$  is a larger perturbation than  $f_i$  and  $f_j$  then  $x_i$  and  $x_j$  exhibit parameter interactions with each other. If the change resulting from a perturbation of two variables is greater when they are perturbed together, as opposed to when they are perturbed individually, we can say that these variables interact with each other. The definition of what constitutes separability and non-separability provided in this section, is the definition we employ for the remainder of this paper.

Related to the notion of additively separable and non-separable problems is the linkage problem (Whitley 1991) associated with Genetic Algorithms and binary representations.

Consider the simple 3-bit deceptive problem in Fig. 4 which shows the allele values and the fitness of each chromosome. In this problem, the chromosome with 111 has the highest fitness. The problem is considered deceptive because a canonical Genetic Algorithm is deceptively guided towards the 000 chromosome even though it is sub-optimal.

We remind the reader that in the context of Evolutionary Algorithms a separable problem can be solved by individually perturbing the additively separable element of the problem, such as  $f(x)$  or  $g(y)$ . In a non-separable problem,  $f(x)$  and  $g(y)$  must be solved simultaneously to find the optimal solution. The deceptive trap problem of Fig. 4 is related to the concept of non-separability because to find the optimal solution a number of mutations would have to occur simultaneously and constructively for each allele. Individual mutations of alleles lead towards the trap. Although mutation is not the only scheme that could help the search, crossover would require the relatively unfit individuals 011, 110, and 101 to be maintained by the population, which is unlikely because they are not deemed to be fit.

|               |               |
|---------------|---------------|
| $f(000) = 28$ | $f(001) = 26$ |
| $f(010) = 22$ | $f(100) = 14$ |
| $f(110) = 0$  | $f(011) = 0$  |
| $f(101) = 0$  | $f(111) = 30$ |

Fig. 4 3-bit deceptive problem

Practitioners in Evolutionary and Genetic Algorithms frequently borrow nomenclature from the biological sciences. One such term is *epistasis*, which is the interaction between genes. In the biological sense, epistasis takes place when the fitness resulting from a gene can be masked by one or more other genes. A definition of epistasis in the field of Genetic Algorithms is “the effect on chromosome fitness of a combination of alleles which is not merely a linear function of the effects of the individual alleles” (Reeves and Wright 1995). This can be related to the deceptive trap problem which exhibits a high degree of epistasis, in that 011 has a low fitness due to the masking effect of 0, and fitness in this instance is clearly not a linear function of the effects of the individual alleles.

Similarly, epistasis in real-valued problems has come to mean the non-separability of the problem, and by definition, the presence of non-linear interactions between parameters. As described by Salomon (1997), “Epistasis describes a nonlinear interaction of parameters with respect to the fitness of an individual.”

It is important to note at this point that even though a function can be non-linear, such as the function  $x^2 + y^2$ , this function is obviously separable and the interactions between parameters are the result of the addition term, which results in a linear parameter interaction. Nonlinear interactions are the result of non-additive operators acting on two or more variables in a function, such as  $xy^2$  or  $x/y$ . From this discussion, it is clear that the concept of epistasis and non-separability (which is the result of a non-linear parameter interaction between sub-functions of a problem) are one and the same.

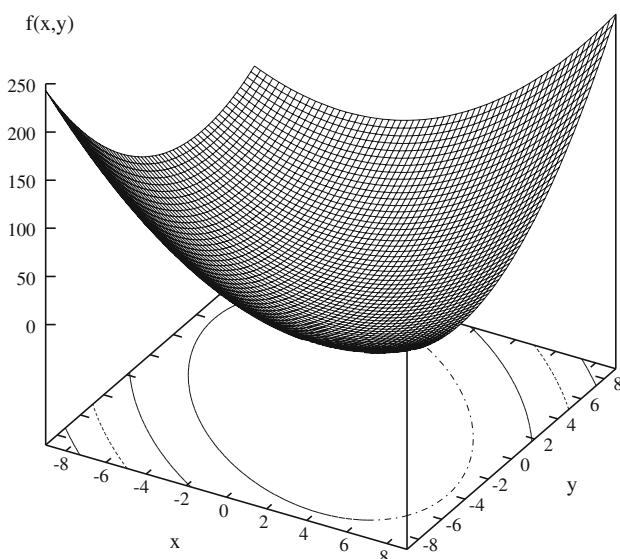
Finally, an interesting observation can be made with respect to how separable problems can be turned into non-separable problems. Consider the ellipsoid function  $F(x, y) = f(x) + f(y)$  where  $f(x) = x^2$  and  $f(y) = a_0y^2$ . This function is separable, but after rotation (which is a linear transformation of the function and does not change its landscape, only its orientation), the ellipsoid function becomes  $F(x, y) = x^2 + a_1xy + a_0y^2$ , a non-separable function with parameter interactions introduced through the term  $xy$ . Note that the term  $a_1$  plays an important role with respect to the degree of parameter interactions. We note that the use of multiplication to introduce parameter interactions is not new; the approach from (Deb et al. 2006) 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, so the experimenter has to be particularly careful to maintain the desired characteristics of the original fitness landscape in each objective. Care must be taken with 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. Similarly, the approach described in Pelikan et al. (2000) introduces epistasis through the multiplication of parameters. The common element between these approaches is that parameter multiplication is responsible for parameter interactions.

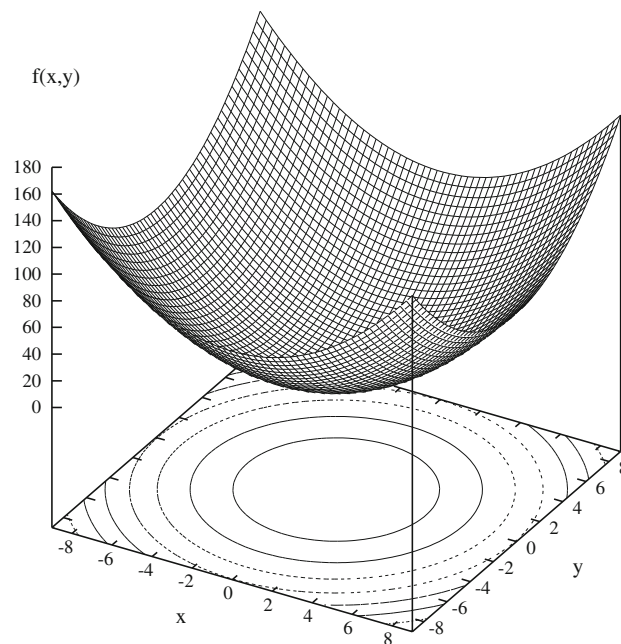
Figure 5 shows the rotated ellipsoid function with  $a_1 = 1.0$ . We note that the value of  $a_1$  is the result of the second-order partial derivative  $F_{xy} = F_{yx} = a_1$ . Clearly the ellipsoid with  $a_1 = 1.0$  is non-separable. Figure 6 shows the ellipsoid function plotted again with  $a_1 = 0.001$ , indicating a high degree of separability. As stated earlier, if the second-order partial derivative is zero then the problem is separable and there are no parameter interactions, and we can see that as  $a_1$  approaches zero the ellipsoid function increases in symmetry and approaches the configuration of a sphere function. Furthermore, the application of a rotation operation in the decision space of a non-linear test problem can be used to introduce non-separability, although clearly it is not the sole means to do so; it just provides a convenient way to do it. Although rotation is not the only means by which parameter interactions can be introduced (they can be introduced with a multiplication operation as we described earlier), it is convenient in that it allows one to explore the performance characteristics of an algorithm without biasing the experiments towards any particular coordinate axes.

#### 4 Rotationally invariant approaches, correlated sampling, and the CSDE: an improved sample-based DE algorithm

In the previous section, we mentioned how rotation can introduce parameter interactions. Now, imagine an



**Fig. 5**  $F(x, y) = x^2 + a_1xy + a_0y^2$  with  $a_1 = 1.0$



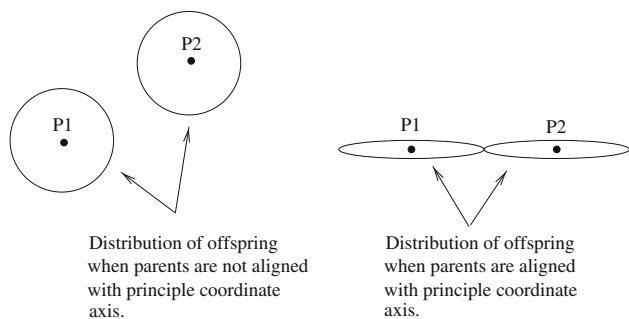
**Fig. 6**  $F(x, y) = x^2 + a_1xy + a_0y^2$  with  $a_1 = 0.001$

arbitrary orientation of the fitness landscape and an algorithm which can find the optimum of such a landscape irrespective of the orientation of the landscape. An algorithm with this property is said to be *rotationally invariant*.

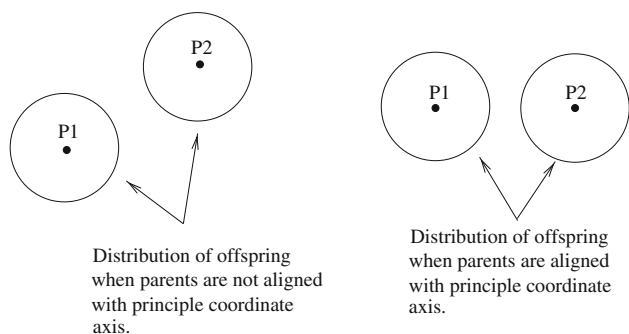
Typically, the rotationally invariant property of an algorithm is related to the distribution of offspring. For example, in Fig. 7, the distribution of offspring clearly changes when the relative position of the parents change, indicating that a non-rotationally invariant operator is used to generate offspring. If the distribution of offspring generated by an evolutionary optimization algorithm varies in relation to the relative orientation of the parents then the algorithm is said to be non-rotationally invariant. A rotationally invariant evolutionary algorithm would produce the same offspring distribution irrespective of the orientation of the parents, as detailed in Fig. 8.

If an optimization algorithm produces better performance when parent solutions are aligned with a particular coordinate axis, then such an algorithm is not rotationally invariant.

Another feature of evolutionary algorithms is whether they are capable of producing correlated samples. For example, in Fig. 9, the uncorrelated sampling scheme samples using a distribution which is independent of the search space being explored by the algorithm. This can result in inefficient exploration of the space. Ideally, in this example, the step sizes along the length of the space should be larger than those along the breadth of the search space. In the correlated sampling scheme, the offspring candidates are efficiently sampled with respect to the extent of the



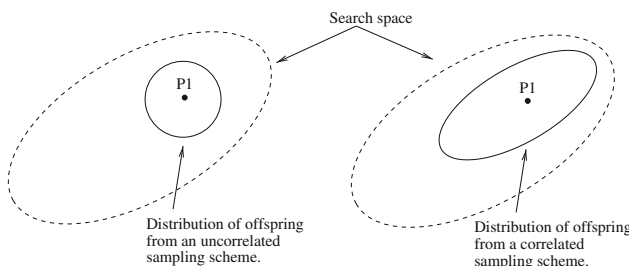
**Fig. 7** Distribution of offspring from a non-rotationally invariant operator



**Fig. 8** Distribution of offspring from a rotationally invariant operator search space, resulting in greater efficiency of the search. As we stated in the introduction, DE has the attractive feature of generating correlated samples.

The characteristic of being rotationally invariant and the ability to produce correlated samples are both critically important features for efficient exploration of the search space.

In this section, we describe the Combinatorial Sampling Differential Evolution (CSDE) algorithm (Li and Iorio 2008) which uses a ‘target’ best individual and maintains diversity using the sampling of difference vectors from two parent vectors. For the purpose of simplicity, we describe the behavior of the algorithm in a two-dimensional decision space, although the process easily generalizes to an arbitrary number of decision space dimensions. We also discuss some of the advantages and characteristics of the approach which are different from the typical DE.



**Fig. 9** Distribution of offspring from correlated and uncorrelated sampling schemes

### 4.1 The CSDE algorithm

The CSDE algorithm works like many other DE algorithms and is described in Algorithm 2. Each individual  $\mathbf{x}^{(i)}$  has an opportunity to participate in the DE calculation. A second individual  $\mathbf{x}^{(r)}$  is chosen for a difference vector calculation such that the population index  $r$  is not equal to  $i$ , and  $r$  is an index randomly chosen from the population.

**Algorithm 2** (CSDE) Non-dominated sorting DE with directed Convergence and Spread which is biased towards generating vectors which are correlated with the search space.

```

1:  $i = \text{floor}(\text{rand}(0, 1] * \text{popsize})$ 
2: for  $k=0$  to  $\text{popsize} - 1$  do
3:   Randomly select  $r \in \{1, 2, \dots, \text{popsize}\}$  such that  $r \neq i$ .
4:    $j = \text{floor}(\text{rand}(0, 1] * N)$ 
5:   if  $\text{rand}(0, 1) \leq \kappa$  then
6:      $C\_Sampling()$ 
7:   else
8:      $UC\_Sampling()$ 
9:   end if
10:   $i = (i + 1) \bmod \text{popsize}$ 
11: end for
    
```

**Algorithm 3**  $C\_Sampling()$

```

1: if  $(\text{rank}(\mathbf{x}^{(i)}) < \text{rank}(\mathbf{x}^{(r)})) \vee (\text{rank}(\mathbf{x}^{(i)}) \equiv \text{rank}(\mathbf{x}^{(r)}) \wedge \text{crowddist}(\mathbf{x}^{(i)}) > \text{crowddist}(\mathbf{x}^{(r)}))$  then
2:   if  $\text{rand}(0, 1) \leq 0.5$  then
3:      $\text{generate\_vector}(\mathbf{x}^{(i)}, \mathbf{x}^{(r)}, \text{flag} = 0)$ 
4:   else
5:      $\text{generate\_vector}(\mathbf{x}^{(i)}, \mathbf{x}^{(r)}, \text{flag} = 1)$ 
6:   end if
7: else if  $(\text{rank}(\mathbf{x}^{(i)}) > \text{rank}(\mathbf{x}^{(r)})) \vee (x^{(i)}_{\text{rank}} \equiv x^{(r)}_{\text{rank}} \wedge \text{crowddist}(\mathbf{x}^{(i)}) < \text{crowddist}(\mathbf{x}^{(r)}))$  then
8:   if  $\text{rand}(0, 1) \leq 0.5$  then
9:      $\text{generate\_vector}(\mathbf{x}^{(r)}, \mathbf{x}^{(i)}, \text{flag} = 0)$ 
10:  else
11:     $\text{generate\_vector}(\mathbf{x}^{(r)}, \mathbf{x}^{(i)}, \text{flag} = 1)$ 
12:  end if
13: else
14:   if  $\text{rand}(0, 1) \leq 0.5$  then
15:    if  $\text{rand}(0, 1) \leq 0.5$  then
16:       $\text{generate\_vector}(\mathbf{x}^{(i)}, \mathbf{x}^{(r)}, \text{flag} = 0)$ 
17:    else
18:       $\text{generate\_vector}(\mathbf{x}^{(i)}, \mathbf{x}^{(r)}, \text{flag} = 1)$ 
19:    end if
20:  else
21:    if  $\text{rand}(0, 1) \leq 0.5$  then
22:       $\text{generate\_vector}(\mathbf{x}^{(r)}, \mathbf{x}^{(i)}, \text{flag} = 0)$ 
23:    else
24:       $\text{generate\_vector}(\mathbf{x}^{(r)}, \mathbf{x}^{(i)}, \text{flag} = 1)$ 
25:    end if
26:  end if
27: end if
    
```

**Algorithm 4** UC\_Sampling()

```

1: for n=0 to N - 1 do
2:   if (rank(x(i)) < rank(x(r))) ∨ (rank(x(i)) ≡ rank(x(r)) ∧
   crowddist(x(i)) > crowddist(x(r))) then
3:     generate_parameter(xj(i), xj(r))
4:     flag = 0
5:   else if (rank(x(i)) > rank(x(r))) ∨ (x(i)rank ≡ x(r)rank ∧
   crowddist(x(i)) < crowddist(x(r))) then
6:     generate_parameter(xj(r), xj(i))
7:     flag = 1
8:   else
9:     if rand(0, 1) <= 0.5 then
10:      generate_parameter(xj(i), xj(r))
11:      flag = 0
12:     else
13:      generate_parameter(xj(r), xj(i))
14:      flag = 1
15:     end if
16:   end if
17:   if uj(i) < xj(L) then
18:     if flag = 0 then
19:       uj(i) = xj(i) + rand(0, 1) * (xj(L) - xj(i))
20:     else
21:       uj(i) = xj(r) + rand(0, 1) * (xj(L) - xj(r))
22:     end if
23:   end if
24:   if uj(i) > xj(U) then
25:     if flag = 0 then
26:       uj(i) = xj(i) + rand(0, 1) * (xj(U) - xj(i))
27:     else
28:       uj(i) = xj(r) + rand(0, 1) * (xj(U) - xj(r))
29:     end if
30:   end if
31:   j = (j + 1) mod N
32: end for

```

**Algorithm 5** generate\_vector(a,b,flag)

```

1: for n=0 to N - 1 do
2:   if flag ≡ 0 then
3:     uj(i) = a + F(a - b)
4:   else
5:     uj(i) = a + F(b - a)
6:   end if
7:   if uj(i) < xj(L) then
8:     uj(i) = a + rand(0, 1) * (xj(L) - a)
9:   end if
10:  if uj(i) > xj(U) then
11:    uj(i) = a + rand(0, 1) * (xj(U) - a)
12:  end if
13:  j = (j + 1) mod N
14: end for

```

**Algorithm 6** generate\_parameter(a,b)

```

1: if rand(0, 1) <= 0.5 then
2:   uj(i) = a + F(a - b)
3: else
4:   uj(i) = a + F(b - a)
5: end if

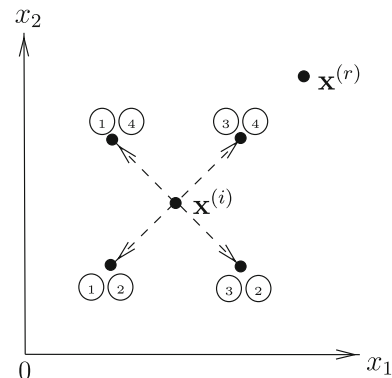
```

Two types of samples are performed in this algorithm around an individual that is deemed to be better than another. The first type of sample we call a C-sample (*correlated sample*), which is outlined in Algorithm 3, such that the vector difference and perturbation are in the same direction, around a better individual (In Fig. 10,  $\mathbf{x}^{(i)}$  is better than  $\mathbf{x}^{(r)}$ , for the purpose of explaining the operation of the algorithm. Of course, if the opposite was true, then sampling would occur around individual  $\mathbf{x}^{(r)}$ . In Algorithms 3 and 4, ‘better’ is described in a multi-objective context, with rank and crowding distance, as expanded upon in Sect. 4.3. For single-objective domains ‘better’ is with respect to a single fitness function evaluation). The point labeled by ① ② corresponds to the point specified by Eqs. 4 and 5. In these equations,  $u_1^{(i)}$  represents the offspring parameter from the DE mutation equation for the first parameter in the decision vector, and  $u_2^{(i)}$  represents the offspring for the second parameter in the decision vector. Similarly, the point labeled by ③ ④ corresponds to the point specified by Eqs. 6 and 7. Both points ① ② and ③ ④ are correlated because they are in the same direction as the difference vector. The points at ① ② and ③ ④ are sampled with the same probability.

$$u_1^{(i)} = x_1^{(i)} + F(x_1^{(i)} - x_1^{(r)}) \tag{4}$$

$$u_2^{(i)} = x_2^{(i)} + F(x_2^{(i)} - x_2^{(r)}) \tag{5}$$

$$u_1^{(i)} = x_1^{(i)} + F(x_1^{(r)} - x_1^{(i)}) \tag{6}$$



**Fig. 10** In a two-dimensional decision space vectors are sampled around a ‘better’ individual

$$u_2^{(i)} = x_2^{(i)} + F(x_2^{(r)} - x_2^{(i)}) \tag{7}$$

The second type of sample is labeled in Fig. 10 by ① ④ and ③ ② which, respectively, correspond to the points generated by Eqs. 4 and 7 and Eqs. 6 and 5. In the CSDE scheme, when Eqs. 4 and 7 are used, then the points at ① ④ can be generated. Similarly, when Eqs. 6 and 5 are used the points at ③ ② can be generated. Figure 10 demonstrates how points are generated depending on which combination of DE equations are used. Both of these samples are uncorrelated and not rotationally invariant because the magnitudes of the difference vectors for these samples is the result of the difference between  $\mathbf{x}^{(i)}$  and  $\mathbf{x}^{(r)}$  and they vary depending on the orientation of  $\mathbf{x}^{(i)}$  and  $\mathbf{x}^{(r)}$ . It is this second type of sampling that contributes diversity to the search. In traditional DE, only a single difference vector can result from two points. Our approach dramatically increases the number of possible samples at the expense of always generating rotationally invariant correlated samples. We call these sample points UC-samples (*uncorrelated and correlated samples*), the generation of which is described in Algorithm 4. UC-Sampling occurs with equal probability for each possible point, including the rotationally invariant correlated sample points ① ② and ③ ④ and the uncorrelated points ① ④ and ③ ②. As the decision space dimension scales, the number of such samples increases in proportion to  $2^D$ , where  $D$  is the decision space dimension. In two decision space dimensions, there are four equations that can specify the possible sample points. In three dimensions, there will be eight equations. This can easily be implemented programmatically by specifying an equal probability for  $u_j^{(i)} = x_j^{(i)} + F(x_j^{(i)} - x_j^{(r)})$  and  $u_j^{(i)} = x_j^{(i)} + F(x_j^{(r)} - x_j^{(i)})$  to be used for each decision space parameter  $j = 1$  to  $D$ , so that all possible samples have an equal chance of occurring.

Whether a C-sample or UC-sample occurs is determined probabilistically by a control parameter  $\kappa$ . This parameter is responsible for controlling the balance between C-sampling and UC-sampling in the generation of offspring.

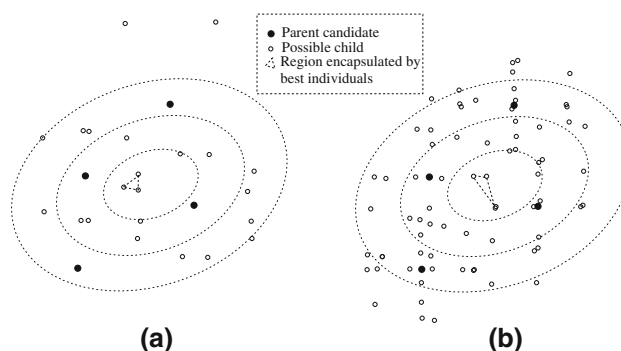
### 4.2 Characteristics and advantages of the CSDE approach

In CSDE, there are two pressures in the generation of offspring; exploitation results from the highly correlated rotationally invariant samples (C-samples) being generated, which rapidly drives the algorithm towards better solutions, and exploration occurs from the UC-sampling, which attempts to discover new and diverse points around the better individual. The UC-sampling method sacrifices emphasis on correlated rotationally invariant points for a dramatic increase in diversity as the decision space

dimension scales to higher dimensions. The general idea of this approach is to increase the diversity that DE is capable of generating using a relatively small population size. A critical point to consider here is that although crossover is also not a rotationally invariant scheme, it only generates points which are aligned with the target parent. The UC-samples are not biased in such a fashion, and although they do not result in rotational invariance, they do produce offspring sample vectors distributed around the target vector.

An attractive feature of CSDE is that the number of potential offspring that can be sampled is bounded by  $2^D$ , as in the crossover based DE described in Fig. 1b. Figure 1a shows how offspring is sampled using our approach. The number of candidates that can potentially be sampled around a base-vector in the sampling based approach is in proportion to  $(Np - 1)2^D$ . From Fig. 1a, CSDE is superior to standard DE with crossover because it can generate points that are not solely sampled along the principle coordinate axes, unlike DE with crossover in Fig. 1b. As a result, CSDE can be highly effective on problems which are non-separable compared with an algorithm which only produces biased samples along the principle coordinate axes aligned with a parent target vector. Furthermore, it bears mentioning that the smallest population size that CSDE can work with is two, unlike rotationally invariant DE/rand/1/bin which requires four individuals.

The difference in efficiency between the CSDE approach and DE/rand/1/bin is detailed further in Fig. 11 where the CSDE is capable of generating offspring in near optimal regions of the search space with far fewer samples than the DE/rand/1/bin approach. Although crossover based DE generates more points, it is apparent from this figure that the sampling based CSDE scheme is superior because of the greater focus it produces on more promising regions of the search space. In contrast, it is clear that crossover based DE/rand/1/bin samples many more offspring but such offspring may not be advantageous towards finding the optima efficiently.



**Fig. 11** Distribution of all possible offspring from 3 parent individuals and a mutation scaling factor  $F = 0.5$ . **a** CSDE and **b** DE/rand/1/bin with crossover

The proposed sampling based approach can be efficiently directed towards more optimal regions using appropriate vector selection. The CSDE approach is also capable of generating more points than standard rotationally invariant DE and like rotationally invariant DE/rand/1/bin, only  $\mathcal{O}(Np)$  processes are required.

#### 4.3 CSDE in a multi-objective domain

In the previous section, we described how CSDE samples vectors near a ‘fitter’ individual. An individual is deemed to be better than another individual with respect to fitness to determine an appropriate direction for the vector difference. If a better direction is not apparent from the measures of fitness associated with both individuals, then a direction is chosen randomly. In the single-objective domain, there is only one measure of fitness to determine if an individual is better than another, but in a multi-objective optimization domain there are multiple conflicting objectives. In order to evaluate the performance of CSDE in a multi-objective domain, the NSGA-II (Deb et al. 2002) framework is employed with DE substituting for the recombination/mutation operations in this algorithm. In the NSGA-II individuals are first compared based on their dominance rank. If individuals are of the same rank, they are compared on their crowding distance. This comparison is described in Algorithms 3 and 4. If after these comparisons, the individuals are deemed to be equivalent, then a random direction for the vector is chosen. The second difference between the single-objective optimization algorithm runs and the multi-objective runs incorporating NSGA-II is that in the single-objective optimization runs if an individual has better fitness than another individual it replaces the inferior individual in the population. In the multi-objective variant of CSDE incorporating NSGA-II, replacement is determined by non-dominated sorting and elitism procedures. We refer the reader to (Deb et al. 2002) for more detailed information on how the NSGA-II is implemented.

## 5 Experiments and methodology

Four DE variants are evaluated in this study. First, a baseline DE technique incorporating three vectors is employed. This baseline approach was briefly discussed in the introductory section. For our purposes, the baseline DE algorithm used here for benchmarking is equivalent to the DE/rand/1/bin approach (Lampinen and Zelinka 2000) because we use it with  $CR = 1.0$  in this study.

Second, the CSDE algorithm is evaluated with  $\kappa$  set to 1.0, 0.5 and 0. When  $\kappa$  is set to 0.5, half the time the algorithm favors C-samples that are highly directed towards better solutions, otherwise it performs UC-sampling. When

$\kappa$  is set to 0 there is no bias, and vectors are sampled using UC-sampling only. In addition, when  $\kappa$  is set to 1.0, the algorithm solely performs C-samples.

A population size of 100 individuals is used for each of the algorithms on each of the test problems for the performance evaluation of the variants over time (unless stated otherwise). For all the DE variants,  $F$  is set to 0.5.

A rotation matrix  $\mathbf{O}$  is used to introduced parameter interactions between decision variables, thereby making the problem non-separable. The same random seed is used for each run of an algorithm on a specific test problem. Rotations for each of these test problems are performed in the decision space, on each plane, using a random uniform rotation matrix, which introduces parameter interactions between all parameters (Iorio and Li 2006). Each algorithm is run 50 times on each single-objective test problem, for a total of 200,000 problem evaluations (unless stated otherwise) for each run to demonstrate a sufficient amount of time is given to the canonical DE algorithm. A new random uniform rotation matrix is generated for each run of each algorithm for the purpose of an unbiased assessment. For the multi-objective test problems, algorithms are run for 50 generations (5,000 problem evaluations) to demonstrate the CSDE is capable of relatively fast performance.

The problems that are used in the evaluation of each of the algorithms in the single objective domain are the rotated Rosenbrock, Griewangk, Ackley and Rastrigin functions. In addition, the Rosenbrock function is evaluated with the algorithm variants to determine sensitivity to population size and scalability in the decision space.

#### 5.1 Multi-objective test problems with parameter interactions

In order to assess the comparative performance of the CSDE on multi-objective problems exhibiting parameter interactions, we use four problems from the literature which are rotated using an orthonormalized rotation matrix which generates randomly uniform rotations for a unit-vector (Iorio and Li 2008).

$$\left. \begin{aligned} f_1(\mathbf{y}) &= y_1 \\ f_2(\mathbf{y}) &= g(\mathbf{y})h(f_1(\mathbf{y}), g(\mathbf{y})) \\ h(f_1(\mathbf{y}), g(\mathbf{y})) &= \exp\left(\frac{-f_1(\mathbf{y})}{g(\mathbf{y})}\right) \\ g(\mathbf{y}) &= 1 + 10(N - 1) + \sum_{i=2}^N [y_i^2 - 10 \cos(4\pi y_i)] \\ \mathbf{y} &= \mathbf{O}\mathbf{x}, -0.3 \leq x_i \leq 0.3, \text{ for } i = 1, 2, \dots, N \\ |f_1| &\leq 0.3 \end{aligned} \right\} \quad (\text{P1})$$

Problem P1 is characterized by a slightly inclined valley in objective  $f_2$ . Objective  $f_1$  is a plane with a gradient sloping

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. This problem was constructed using the ZDT framework. The  $g$  function specifies the modalities of the problem, and the  $h$  function specifies the shape of the Pareto-optimal front. In Problem P1, the decision space is subject to a rotation matrix  $\mathbf{O}$ .

Three further rotated problems which are also described in Iorio and Li (2008) will be used in this study. Problem P2 has a Pareto-optimal front which is not continuous, as specified by the  $h$  function.  $f_1$  is bounded to the range,  $|f_1| \leq 1.0$ , 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.

$$\left. \begin{aligned} f_1(\mathbf{y}) &= y_1 \\ f_2(\mathbf{y}) &= g(\mathbf{y})h(f_1(\mathbf{y}), g(\mathbf{y})) \\ h(f_1(\mathbf{y}), g(\mathbf{y})) &= 1.0 + \exp\left(\frac{-f_1(\mathbf{y})}{g(\mathbf{y})}\right) \\ &+ \left(\frac{f_1(\mathbf{y}) + 1.0}{g(\mathbf{y})}\right) (\sin(5\pi f_1(\mathbf{y}))) \\ g(\mathbf{y}) &= 1 + 10(N - 1) + \sum_{i=2}^N [y_i^2 - 10 \cos(\pi y_i)] \\ \mathbf{y} &= \mathbf{Ox}, \quad -1.0 \leq x_i \leq 1.0, \text{ for } i = 1, 2, \dots, N \\ |f_1| &\leq 1.0 \end{aligned} \right\} \quad (\text{P2})$$

Problem P3 has decision space variables which increment at a regular interval, evaluate with non-regular 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 towards lower  $f_1$  values, making it difficult to find solutions in this region.  $f_1$  is bounded to the range,  $0.3 \leq f_1 \leq 1.0$ , to guarantee that the values of  $f_1$  evaluate consistently no matter what rotation operation the decision space vector is subjected to. Plots of the objective space are not shown as the objective space is unchanged under a rotation of the decision space.

$$\left. \begin{aligned} f_1(\mathbf{y}) &= 1.0 - \exp(2.0y_1)\sin^6(6\pi y_1)/9.0 \\ f_2(\mathbf{y}) &= g(\mathbf{y})h(f_1(\mathbf{y}), g(\mathbf{y})) \\ h(f_1(\mathbf{y}), g(\mathbf{y})) &= 1.0 - \left(\frac{f_1(\mathbf{y})}{g(\mathbf{y})}\right)^2 \\ g(\mathbf{y}) &= 1 + 10(N - 1) + \sum_{i=2}^N [y_i^2 - 10 \cos(\pi y_i)] \\ \mathbf{y} &= \mathbf{Ox}, \quad -1.0 \leq x_i \leq 1.0, \text{ for } i = 1, 2, \dots, N \\ 0.3 &\leq f_1 \leq 1.0 \end{aligned} \right\} \quad (\text{P3})$$

Problem P4 is based on the Schwefel function, and is multi-modal and highly deceptive with a local Pareto-

optimal front which is close to the global Pareto-optimal front. In the decision space the local Pareto-optimal region is actually located far from the Pareto-optimal set in the decision space.

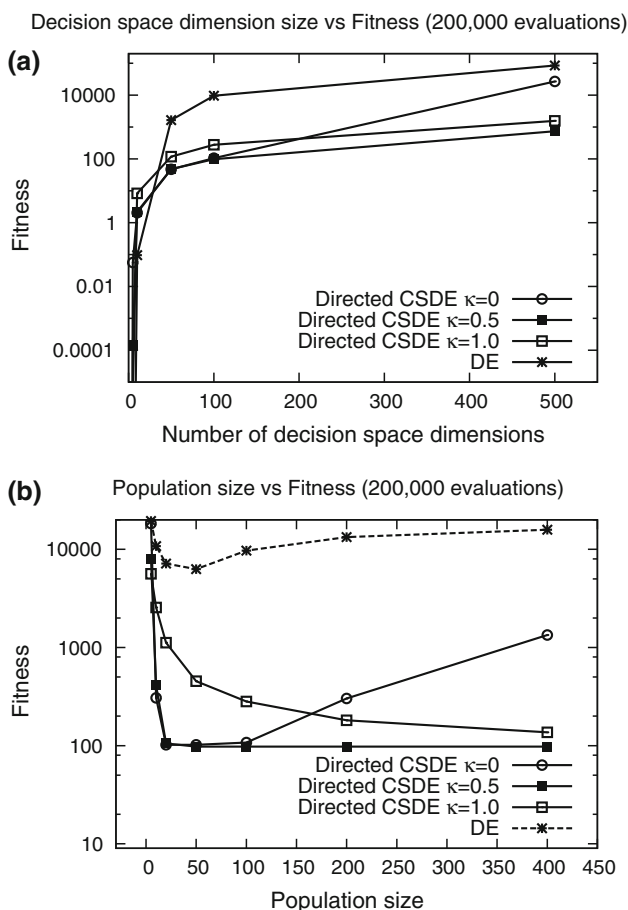
$$\left. \begin{aligned} f_1(\mathbf{y}) &= y_1 \\ f_2(\mathbf{y}) &= g(\mathbf{y})h(f_1(\mathbf{y}), g(\mathbf{y})) \\ h(f_1(\mathbf{y}), g(\mathbf{y})) &= \exp\left(\frac{-f_1(\mathbf{y})}{g(\mathbf{y})}\right) \\ g(\mathbf{y}) &= 1.0 + 0.015578(N - 1.0) \\ &+ \sum_{i=2}^N (y_i^2 - 0.25(y_i \sin(32.0\sqrt{|y_i|}))) \\ \mathbf{y} &= \mathbf{Ox}, \quad -1.0 \leq x_i \leq 1.0, \text{ for } i = 1, 2, \dots, N \\ |f_1| &\leq 1.0 \end{aligned} \right\} \quad (\text{P4})$$

### 5.2 Measuring algorithm performance on multi-objective problems

A critical question arises when one is dealing with multi-objective problems, and that is how can one measure their performance? It is apparent that there are many different performance metrics reported in the literature, some of which are used more frequently than others. Furthermore, some performance metrics have highly desirable theoretical qualities. The first of these that we report on is the hyper-volume indicator. As proposed by Zitzler and Thiele (1998), the fundamental idea of this approach is that the larger the area the solutions can cover in the objective space, the better the approximation set is. It measures the size of the area that is dominated by the boundary resulting from the approximation set in the objective space. It has the attractive feature of not requiring a known Pareto-optimal set to evaluate the quality of the approximation set. The hyper-volume indicator is particularly attractive for practitioners interested in evaluating approximation sets because whenever one approximation set dominates another approximation set, the hyper-volume indicator always yields a better value for the dominating set (Fleischer 2003). Due to its desirable characteristics and its wide use by practitioners, we have used it in this study.

## 6 Results on single and multi-objective test problems

From Fig. 12a, it is apparent that CSDE with  $\kappa = 0.5$  is insensitive to an increase in decision space size on the 100 dimensional rotated Rosenbrock function. In addition, it is able to find highly competitive solutions which are far superior to a canonical rotationally invariant DE/rand/1/bin approach, which performed poorly. In addition, in Fig. 12b one can see that the CSDE approach with  $\kappa = 0.5$  is highly insensitive to a change in population size and is capable of



**Fig. 12** Problem dimensionality vs. fitness and population size vs. fitness after 200,000 evaluations on the rotated Rosenbrock function

finding similarly good solutions after 200,000 evaluations because of the order of magnitude higher degree of sampling that is possible compared with the canonical rotationally invariant DE/rand/1/bin. In contrast, the performance of CSDE with  $\kappa = 0$  where only U-sampling is performed, peaks in performance between a population size of 20–100 individuals on the rotated Rosenbrock function in 100 dimensions. This indicates that a large population size detracts from the performance of CSDE when U-sampling is used. The reasons for this are that as the number of individuals in the population increases, the probability of sampling highly directed correlated samples reduces when U-sampling is employed. Rotationally invariant correlated sampling is clearly beneficial to the performance of the CSDE approach, to make it more insensitive to the population size. In contrast, the CSDE approach with  $\kappa = 1.0$  is highly dependent on population size for sampling diversity, and the performance only begins to approach CSDE with  $\kappa = 0.5$  as the population size approaches 500 individuals on this problem. It is also clear from Fig. 12 that rotationally invariant DE/rand/1/bin

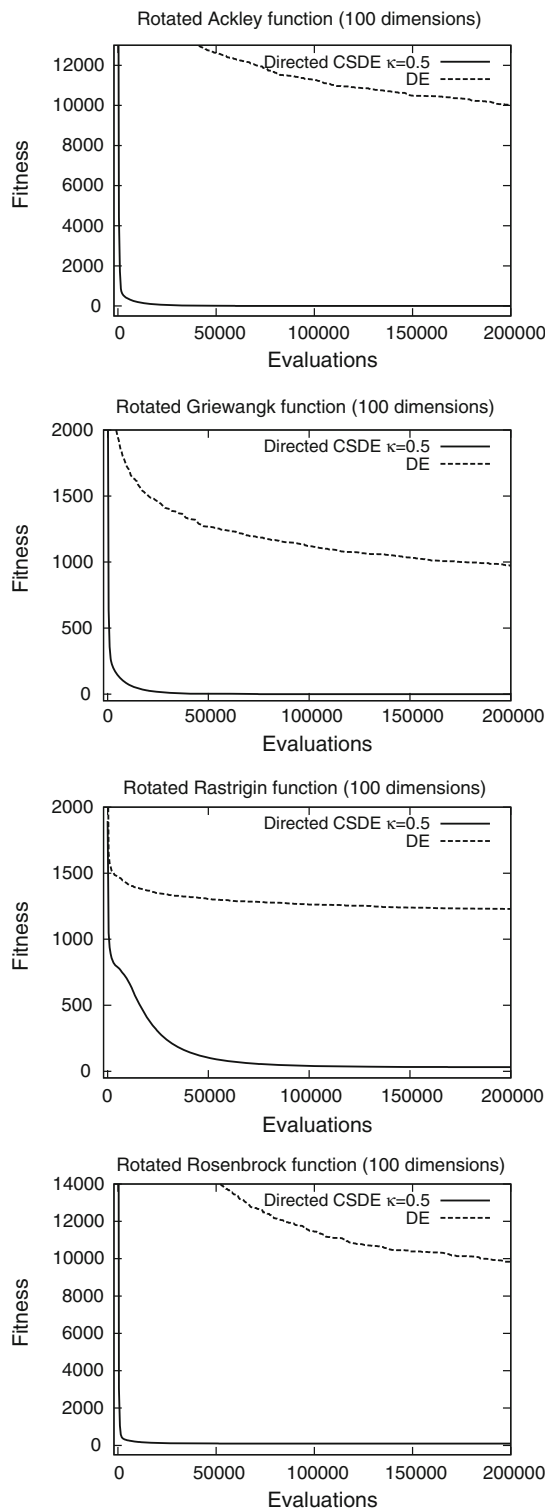
performs extremely poorly on the 100 dimensional Rosenbrock problem in the presence of parameter interactions.

From these results, it is clearly apparent that when  $\kappa = 0.5$ , the CSDE algorithm has superior performance over the UC-sampling approach which uses  $\kappa = 0$ . This indicates that sampling of highly directed rotationally invariant correlated vectors is critically important for the algorithm to remain insensitive to population size variations as well as discover highly fit solutions in extremely large decision spaces.

In order to test the performance of the variants over time, the rotated Ackley, Rastrigin, Griewangk and Rosenbrock functions were employed with 100 dimensions. The results in Fig. 13 indicate that the performance of the CSDE approach with  $\kappa = 0.5$  are dramatically superior to the rotationally invariant DE/rand/1/bin algorithm which does not employ sampling.

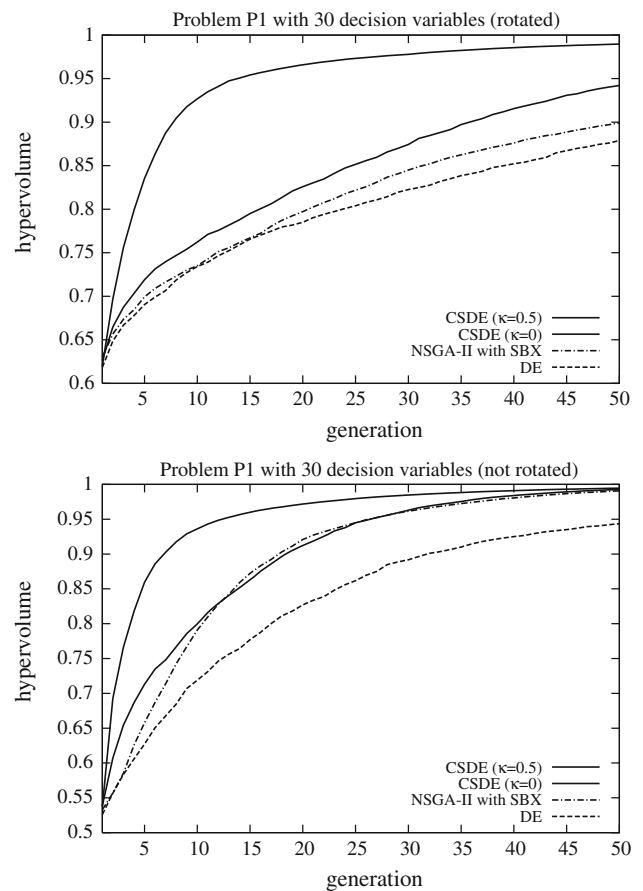
In Figs. 14, 15, 16, and 17, the average hyper-volume indicator value over 50 generations is presented on problem P1, P2, P3, and P4 respectively in both the rotated (non-separable) and un-rotated (separable) case. Furthermore, from these results one can see that NSGA-II with SBX becomes competitive with CSDE ( $\kappa = 0.5$ ) when the problems are un-rotated after about 50 generations. When the problems are rotated, NSGA-II with SBX is no longer competitive with CSDE ( $\kappa = 0.5$ ). Furthermore, even the UC-sampling approach in CSDE with  $\kappa = 0$  is competitive with NSGA-II in the rotated cases. The baseline DE approach demonstrated the worse performance in both the rotated and un-rotated cases. In addition, when  $\kappa = 0.5$  CSDE has superior performance over CSDE with  $\kappa = 0$ . This indicates that highly directed rotationally invariant correlated samples in addition to vectors which contribute to diversity is critically important for the algorithm to make efficient progress.

In order to test the scalability of the CSDE approach, Problem P1 is optimized in 200 decision space dimension (Fig. 18). The relative performance of each of the rotationally invariant DE algorithms is similar in the rotated and un-rotated case. NSGA-II demonstrates that it has significant difficulties in such a high decision space dimension in the presence of parameter interactions. Interestingly, the baseline DE is the second best performing approach next to the CSDE ( $\kappa = 0.5$ ). One would expect that the much higher degree of sampling in CSDE ( $\kappa = 0$ ) would demonstrate better performance than the baseline DE. The obvious explanation is that although the CSDE ( $\kappa = 0$ ) which performs UC-sampling provides many samples, the vast majority of the samples are not rotationally invariant nor correlated with the search space unlike the baseline DE. When  $\kappa = 0.5$  half the time rotationally invariant correlated and highly directed C-sampling is performed, improving the



**Fig. 13** Fitness over 200,000 evaluations on rotated problems in 100 dimensions of the decision space for the rotated Ackley, Griewangk, Rastrigin and Rosenbrock functions

performance of CSDE. The result reported here is consistent with the observation that biasing selection pressure can improve the performance of differential evolution on non-



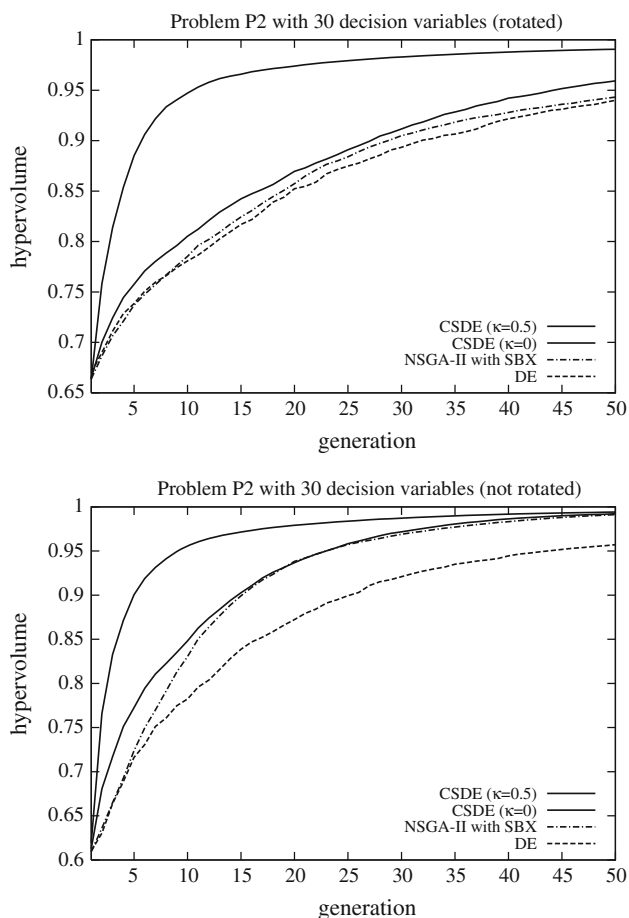
**Fig. 14** Average hyper-volume over successive generations on problem P1 in 30 decision space dimensions

separable problems (Sutton et al. 2007). Whether the bias is a result of favoring fitter solutions for selection, or biasing the direction of the search as is the case with our approach, both are useful for improving the performance of rotationally invariant DE on difficult non-separable optimization problems.

### 7 Multi-objective truss-topology optimization problems

The truss problems we attempt to optimize in this paper have parameter interactions between some, but not all of the parameters. Many models that approximate problems in the real world are of this type, and the purpose of this case study is to investigate and evaluate the performance of the CSDE on such a problem.

In this paper, we address the question of whether or not the CSDE is efficient and competitive in comparison with a multi-objective genetic algorithm applied to a multi-objective truss-topology optimization problem. The purpose of the study presented in this chapter is to convincingly demonstrate that the CSDE can also be practically applied to real

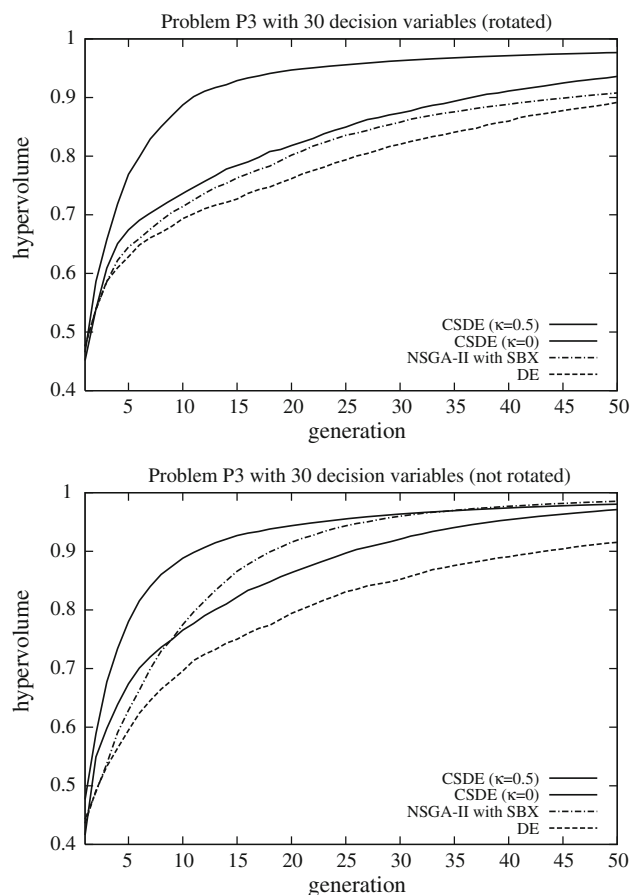


**Fig. 15** Average hyper-volume over successive generations on problem P2 in 30 decision space dimensions

world problems, such as the plane truss optimization problem, and demonstrate its competitiveness with NSGA-II incorporating the SBX operator with polynomial mutation. Furthermore, we describe the truss-topology optimization problem in detail so that practitioners can implement the FEM (Finite Element Model) we have employed in this study.

The multi-objective optimization of truss structures has received attention from both the structural engineering community, and from evolutionary optimization practitioners (Ruy and Yang 2001; Coello and Christiansen 2000; Deb et al. 2000). In truss-structure design optimization, a number of objectives can be considered, such as the minimization of stresses in the truss, minimization of the weight, and the minimization of displacement of truss nodes. These objectives are sometimes formulated as a single objective problem with a constraint, or as a multi-objective problem.

In the following study, given a model of a truss, the optimization task is to find the Pareto-optimal set of trade off solutions with respect to the two objectives: minimizing

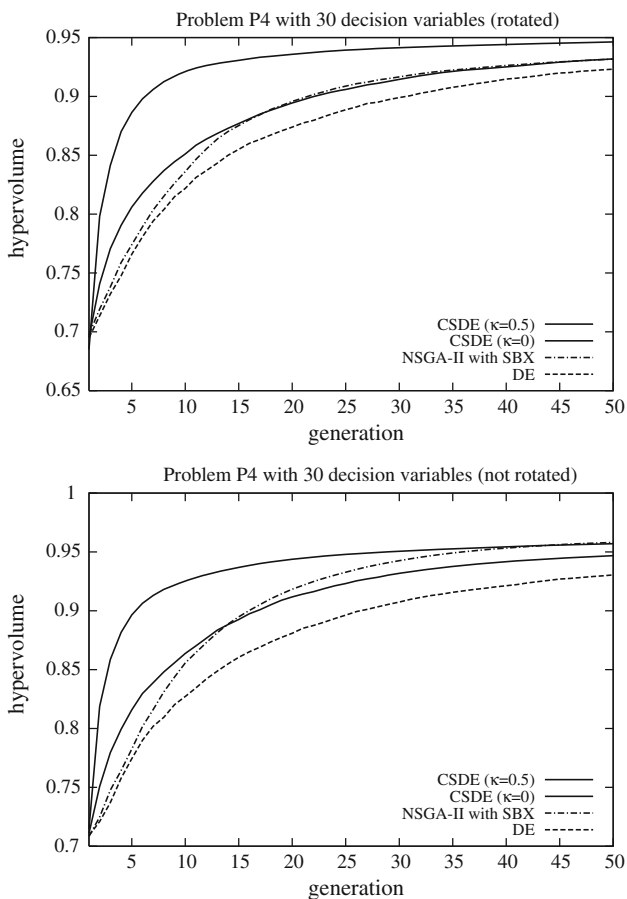


**Fig. 16** Average hyper-volume over successive generations on problem P3 in 30 decision space dimensions

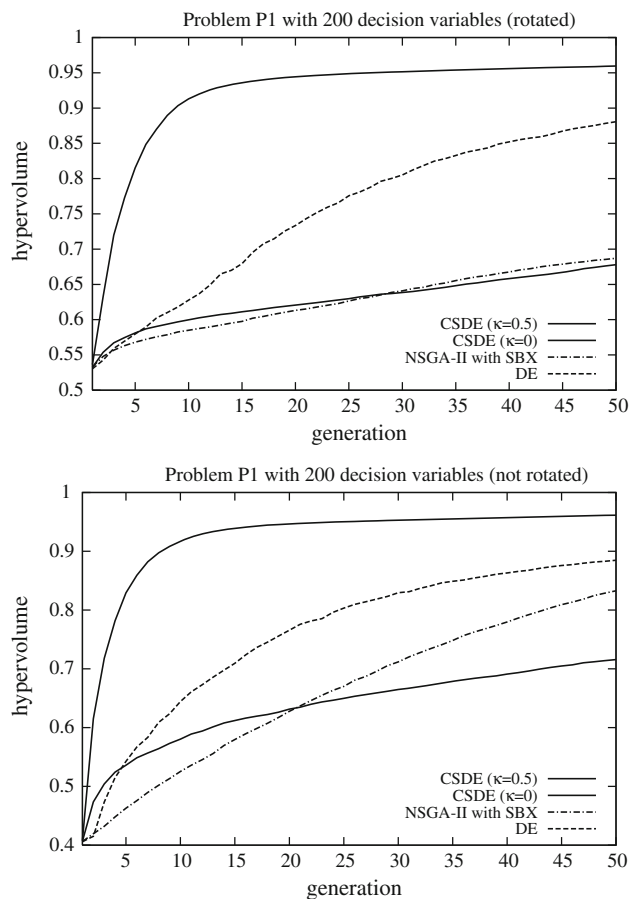
the worst displacement of any node in the truss, and minimizing the weight of the truss.

An approach for modeling truss structures for the purpose of multi-objective optimization was described in Deb et al. (2000), which builds upon an approach applied to single objective optimization (Deb et al. 1998). In the multi-objective approach, truss structures are minimized with respect to weight and displacement of nodes. These objectives are conflicting, in that lighter trusses can be expected to have greater node displacement, and vice versa. Furthermore, truss topology and member cross section areas are evolved.

A truss structure is composed of a number of bars connected to each other at node points. The node points that are required are called *basic nodes*, and node points that may or may not be present in the truss are termed *non-basic nodes*. Typically, the *basic nodes* are the support nodes and the load bearing nodes. In the case of Fig. 19, the support nodes are ① and ②, and the load bearing nodes are ③ and ⑤. All nodes are also numbered in an enclosed circle, and the bar numbers are not circled.



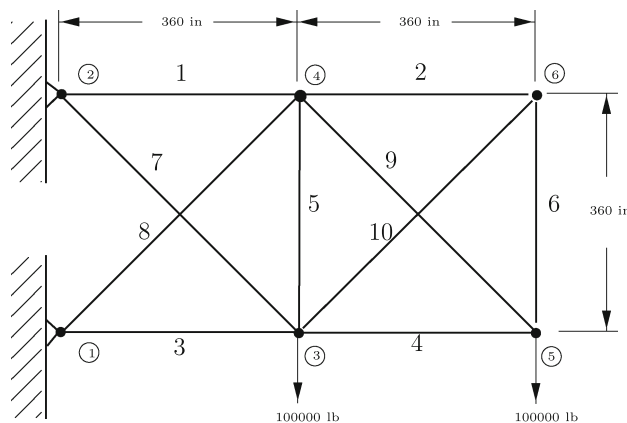
**Fig. 17** Average hyper-volume over successive generations on problem P4 in 30 decision space dimensions



**Fig. 18** Average hyper-volume over successive generations on problem P1 in 200 decision space dimensions

Each bar in the truss has a cross-sectional area associated with it. Each of the members in the truss are identified and numbered (Fig. 19). The cross-sectional areas are a vector of real-values, one for each bar, which undergoes evolution for the purposes of finding more optimal values for these areas. The bars are initialized within the range of  $-A$  to  $A$ . Bars that have cross-sectional areas less than some critical value  $\epsilon_1$ , are not included in the realized truss structure, or any calculations to determine the forces and displacements of the realized truss structure. The possibility of the cross-sectional area of each bar taking on values less than  $\epsilon_1$  provides a means of discovering a variety of different topologies for the truss. For example, Fig. 20 shows bars 2, 5, 6, and 10, absent from the realized truss structure.

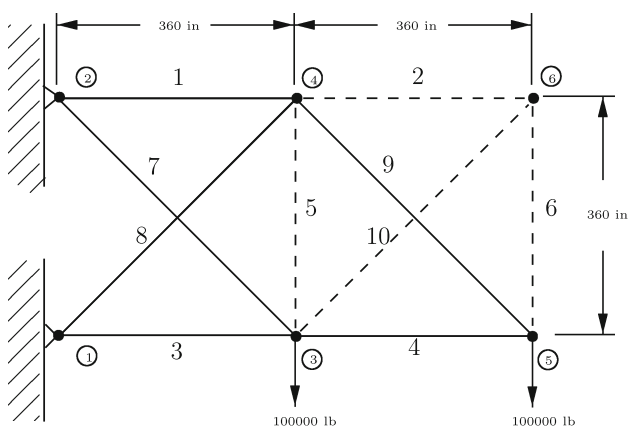
Most models of problems we find in the real-world are constrained problems, and the plane-truss structural optimization problem is no exception. In order to implement this problem, a number of constraints must be calculated. For the truss topology and shape design problem, three types of constraints are considered: Constraint-I specifies the degree to which the truss is acceptable to the user,



**Fig. 19** The 10 bar truss problem with all bars present

Constraint-II specifies whether the truss is kinematically stable or not, and Constraint-III deals with tension in the truss structure.

*Constraint-I* The first constraint is concerned with whether or not the realized truss structure is acceptable to the user. A structure is acceptable if all basic nodes are

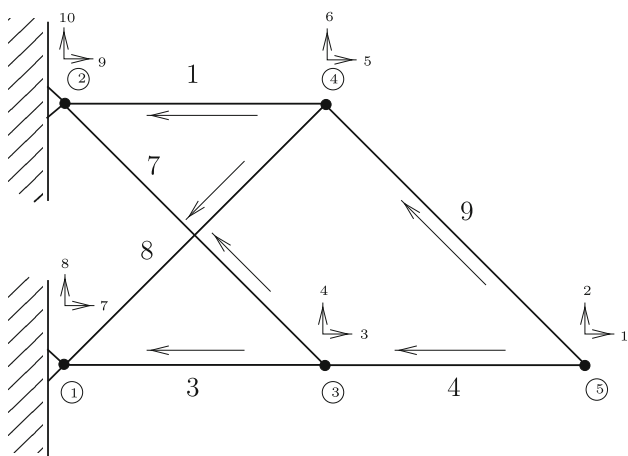


**Fig. 20** The 10 bar truss problem with bars 2, 5, 6, and 10 absent from the realized truss

present in the realized truss structure (Eq. 8), where *NoBN* is *Number of Basic Nodes* and *NoBNP* is *Number of Basic Nodes Present*. The presence of basic nodes can be discovered with a simple breadth first search of the truss structure. If constraint  $c_1$  is violated, no further calculations occur.

$$c_1 = -(10e15) \cdot (NoBN) - (NoBNP) \tag{8}$$

**Constraint-II** The second constraint is concerned with determining if the truss is *kinematically stable*. This is achieved with a *Singular Value Decomposition* of the stiffness matrix. Before this can occur, the *stiffness matrix* needs to be computed, First, an automatic scheme for numbering the degrees-of-freedom, of each node in the realized truss is used. This numbering scheme can be started from an arbitrary node in the structure, but should be applied consistently (see Fig. 21). The basic load bearing nodes have their degrees-of-freedom numbered first, followed by the nodes with unknown displacements.



**Fig. 21** The 10 bar truss problem with bars 2, 5, 6, and 10 absent from the realized truss and with the degrees-of-freedom numbered

Finally, the degrees-of-freedom of the support nodes are numbered as in Fig. 21. This scheme for numbering the degrees-of-freedom of the realized truss simplifies the calculation of displacements at each node. At this stage, the *stiffness matrix* of the realized truss is computed using the *Direct Stiffness Method* (Hibbeler 2006). Each of the elements in the realized truss has to be transformed from its local coordinates to global coordinates. Equations 9 and 10 are used to calculate the *direction cosines* for a member in the realized truss.

$$\lambda_{x_i} = \cos \theta_{x_i} = \frac{x_{F_i} - x_{N_i}}{L_i} \quad i = 1, \dots, m \tag{9}$$

$$\lambda_{y_i} = \cos \theta_{y_i} = \frac{y_{F_i} - y_{N_i}}{L_i} \quad i = 1, \dots, m \tag{10}$$

$x_{F_i}$  is the  $x$ -coordinate of the far end of member  $i$ , and  $x_{N_i}$  is the  $x$ -coordinate of the near end of member  $i$ , where there are  $m$  members in the realized truss. These direction cosines are used in the global stiffness matrix  $\mathbf{k}_i$ . In this matrix calculation, the global stiffness matrix,  $\mathbf{k}_i$ , is calculated for each member  $m$  in the realized truss. The indices of the global stiffness matrix are numbered, and correspond to the degrees-of-freedom of the truss. In Fig. 21, the arrows along each member show the direction from the near to far ends of a member, and each node is numbered with its degrees-of-freedom. In this example,  $N_{x_3}$  is 4,  $N_{y_3}$  is 3,  $F_{x_1}$  is 7, and  $F_{y_1}$  is 8.  $E$  is Young's Modulus,  $A_i$  is the cross sectional area of member  $i$ , and  $L_i$  is the length of member  $i$ .

$$\mathbf{k}_i = \frac{A_i E}{L_i} \begin{matrix} N_{x_i} & N_{y_i} & F_{x_i} & F_{y_i} \\ \begin{bmatrix} \lambda_{x_i}^2 & \lambda_{x_i} \lambda_{y_i} & -\lambda_{x_i}^2 & -\lambda_{x_i} \lambda_{y_i} \\ \lambda_{x_i} \lambda_{y_i} & \lambda_{y_i}^2 & -\lambda_{x_i} \lambda_{y_i} & -\lambda_{y_i}^2 \\ -\lambda_{x_i}^2 & -\lambda_{x_i} \lambda_{y_i} & \lambda_{x_i}^2 & \lambda_{x_i} \lambda_{y_i} \\ -\lambda_{x_i} \lambda_{y_i} & -\lambda_{y_i}^2 & \lambda_{x_i} \lambda_{y_i} & \lambda_{y_i}^2 \end{bmatrix} \end{matrix}$$

If a truss has  $m$  members, it has  $2m$  degrees of freedom, and the global stiffness matrix of the truss is  $2m$ -by- $2m$ . The global stiffness matrix of the entire truss is calculated in Eq. 11, by summing each of the global stiffness matrices of each member.

$$\mathbf{K} = \sum_{i=1}^m \mathbf{k}_i \tag{11}$$

The *stiffness matrix*  $\mathbf{K}$  has a useful property in that, if the matrix is *positive definite*, then the truss that the *stiffness matrix* described is a structure. If the *stiffness matrix* is not *positive definite*, and the corresponding truss is a mechanism. A mechanism is different from a structure, in that it is capable of movement as a result of how its joints are connected. A structure is not a mechanism because its joints are configured in such a way that natural movement is impossible. Obviously, we are only interested in trusses

which do not have the behavior of a mechanism. Before the positive definiteness of the stiffness matrix can be determined, the global stiffness matrix  $\mathbf{K}$  is partitioned as in Eqs. 13, 14 and 15.  $\mathbf{D}_k$  are the known displacements of  $\mathbf{D}$ , and  $\mathbf{Q}_k$  are the known forces of  $\mathbf{Q}$ . Similarly,  $\mathbf{D}_u$  are the unknown displacements of  $\mathbf{D}$ , and  $\mathbf{Q}_u$  are the unknown forces of  $\mathbf{Q}$ . Typically, the known displacements are  $\mathbf{0}$ , and Eq. 14 can be simplified to Eq. 16. Equation 12 describes the relationships between  $\mathbf{Q}$ ,  $\mathbf{K}$ , and  $\mathbf{D}$ . For the purposes of the truss optimization problem, we only need to solve Eq. 16 to find the displacement objective that requires minimizing.

The matrix  $\mathbf{K}_{11}$  is decomposed using singular value decomposition (Golub–Reinsch SVD algorithm) into  $\mathbf{K}_{11} = \mathbf{USV}^T$ , where  $\mathbf{U}$  is an  $2m$ -by- $2m$  orthogonal matrix,  $\mathbf{S}$  is an  $2m$ -by- $2m$  diagonal matrix of singular values, and  $\mathbf{V}^T$  is the transpose of a  $2m$ -by- $2m$  matrix. Equation 16 is then solved for the unknown displacements,  $\mathbf{D}_u$ , using the singular value decomposition  $\mathbf{U}, \mathbf{S}, \mathbf{V}$  and the known forces,  $\mathbf{Q}_k$ .

The SVD technique is used in the solution of the linear equation problem to avoid issues associated with ill-conditioned matrices. The diagonal matrix,  $\mathbf{S}$ , is the squares of the eigen-values of  $\mathbf{K}_{11}$ . If some of the eigen-values are zero, or close to zero by some very small value  $\epsilon_2$ , then the matrix is not positive definite. For a matrix to be positive definite, its eigen-values must be positive. We use this property for the stability constraint, multiplying the number of zero/near-zero eigen values by some large penalty, and thereby determine whether the truss is kinematically stable, or the degree of instability present in the truss. The eigen-values are the diagonal of the matrix  $\mathbf{S}$ , represented by  $S_{ii}$ . If the truss is unstable according to the constraint  $c_2$ , then the displacements calculated are not physically meaningful and no further calculations will occur.

$$\mathbf{Q} = \mathbf{KD} \tag{12}$$

$$\begin{bmatrix} \mathbf{Q}_k \\ \mathbf{Q}_u \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{D}_k \\ \mathbf{D}_u \end{bmatrix} \tag{13}$$

$$\mathbf{Q}_k = \mathbf{K}_{11}\mathbf{D}_u + \mathbf{K}_{12}\mathbf{D}_k \tag{14}$$

$$\mathbf{Q}_u = \mathbf{K}_{21}\mathbf{D}_u + \mathbf{K}_{22}\mathbf{D}_k \tag{15}$$

$$\mathbf{Q}_k = \mathbf{K}_{11}\mathbf{D}_u \tag{16}$$

*Constraint-III* The final constraint (Eq. 17) calculates the tension and compression in each bar of the truss. If the tension and compression in any bar exceeds the maximum allowed tension and compression in that bar, then a bracket penalty operator  $\langle \rangle$  is used (Eq. 18), and no further calculation occurs. The stress in each member of the realized truss can be calculated as in Eq. 17, where  $E$  is Young’s Modulus for the material the bar is composed of, and  $L_i$  is the length of the bar. If the stress  $T_i$  is positive the member  $i$  is in tension, and if it is negative the member is

in compression. Constraint  $c_3$  is concerned with eliminating violations of the maximum allowed tensional and compressional stress, denoted by  $\sigma$ .

$$T_i = \frac{E}{L_i} \begin{bmatrix} -\lambda_{x_i} & -\lambda_{y_i} & \lambda_x & \lambda_y \end{bmatrix} \begin{bmatrix} D_{N_{x_i}} \\ D_{N_{y_i}} \\ D_{F_{x_i}} \\ D_{F_{y_i}} \end{bmatrix} \quad i = 1, \dots, m \tag{17}$$

$$c_3 = -(10e5) \cdot \sum_{i=1}^m \left\langle \frac{\sigma}{|T_i|} - 1.0 \right\rangle \tag{18}$$

It is not until each of the constraints have been passed that the objectives are calculated. The first objective calculates the weight of the realized truss, and the second objective is concerned with the worst displacement of any node in the realized truss.

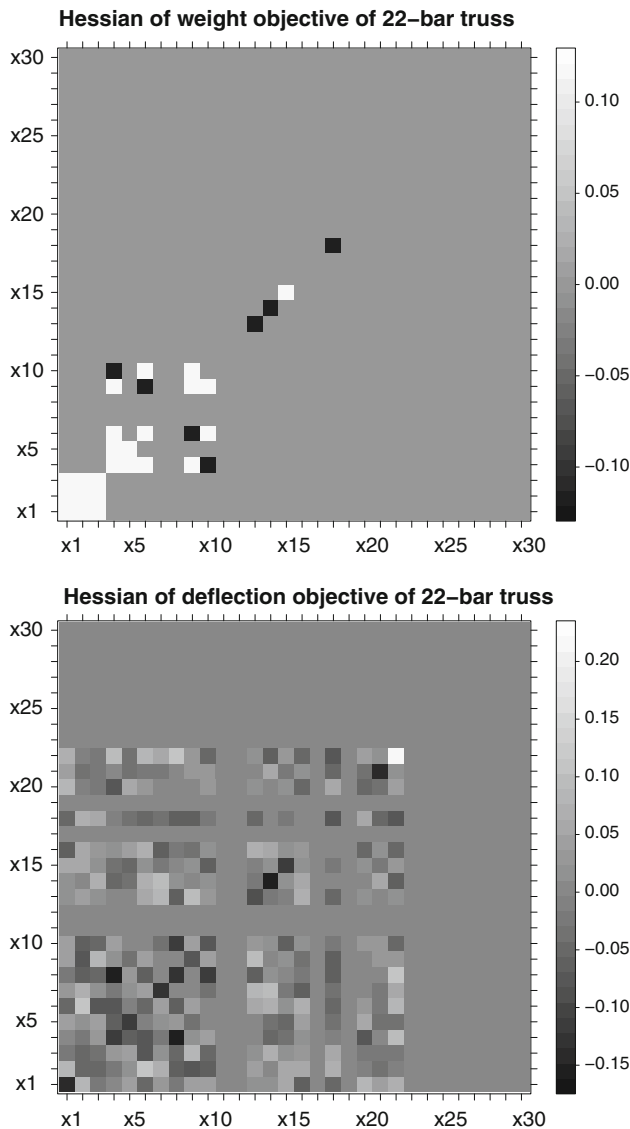
*Objective-I* The weight of the truss is calculated according to Eq. 19. The weight of each bar in the realized truss is calculated using the known density ( $\rho$ ), length, and cross-sectional area of each member. The summed weights is the total weight of the truss.

$$W = \sum_{i=1}^m A_i L_i \rho \tag{19}$$

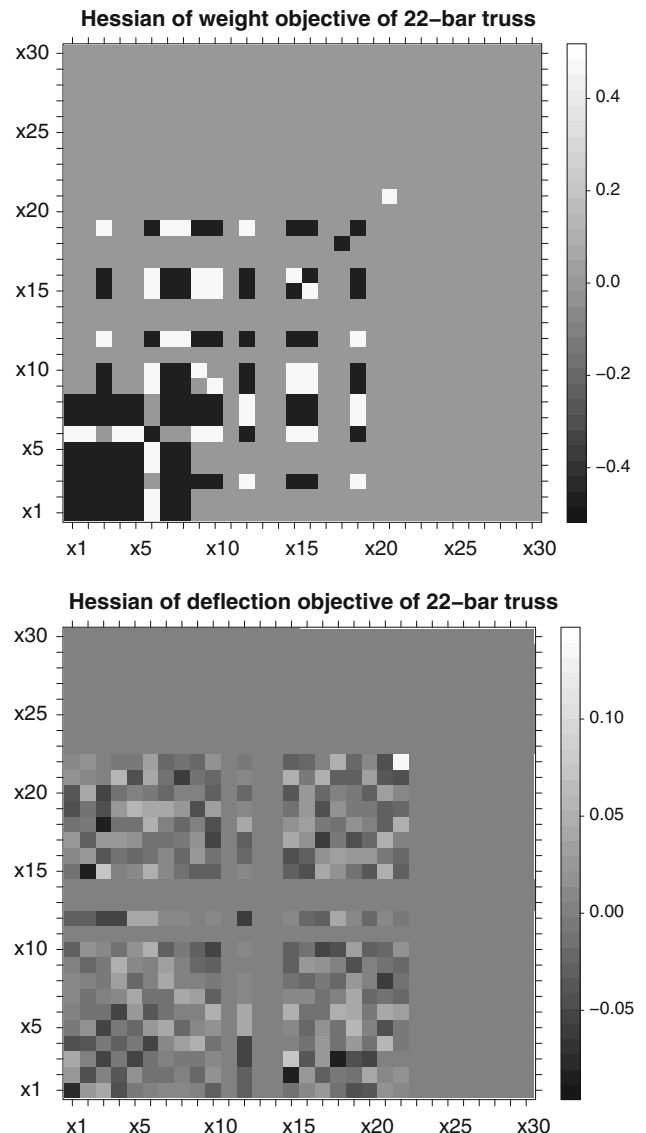
*Objective-II* The worst displacement can be acquired from  $\mathbf{D}_u$ , the vector of unknown displacements in the structure. This may be the worst displacement out of all truss nodes, or it may be the worst displacement of a specific truss node in either  $x$  or  $y$  axis, depending on the requirements of the particular problem.

The truss design and shape optimization problems are problems that exhibit parameter interactions between decision variables in the tail end of the Pareto-optimal front (Fig. 23), where the weight minimization objective Hessian of a tail end solution demonstrates larger relative partial-second order derivatives than the Hessian associated with the weight minimization objective, where parameter interactions are significantly less (Fig. 22). The first truss problem used in this study is the 13-bar problem, which is a shape and topology optimization problem. With respect to this problem, an optimization algorithm has to learn the coordinates of truss nodes, as well as appropriate cross-sectional areas for truss bars. In total, there are 17 decision variables in this problem which have to be optimized. The 22-bar problem is similar to the 13-bar problem, in that it too is a topology and shape optimization problem with 22 bars in the base structure, and 30 decision variables in total.

The CSDE ( $\kappa = 0$ ), and CSDE ( $\kappa = 0.5$ ) and the NSGA-II with simulated binary crossover (SBX), will be evaluated on two truss problems. The scaling factor  $F$ , for each of the DE variants employed, was set to 0.5 for CSDE



**Fig. 22** The 22-bar truss problem Hessian matrix for a parameter space vector that maps to a weight of 3,101.70 pounds and a deflection of 9.27 inches



**Fig. 23** The 22-bar truss problem Hessian matrix for a parameter space vector that maps to a weight of 10,482.56 pounds and a deflection of 4.78 inches

( $\kappa = 0$ ) and CSDE ( $\kappa = 0.5$ ), as it was for all previous experiments. The SBX variant used a mutation rate of  $1/N$  and a crossover rate of 0.9.  $\eta_c$  and  $\eta_m$  are parameters within the NSGA-II with SBX that control the distribution of the crossover and mutation probabilities and were assigned values of 10 and 50, respectively. In all the algorithm variants used on the truss problems the constraint domination principle is employed.

### 7.1 13-Bar truss topology and shape design problem

A 13-bar truss topology and shape optimization problem (Espi 1998) which builds on the simple 10-bar truss problem (Rajeev and Krishnamoorthy 1997) is also used by

practitioners to evaluate optimization algorithms. From Fig. 24, it is apparent that the 13-bar problem has two load bearing nodes ③ and ⑤, and two support nodes, ① and ②. Unlike the 10-bar truss problem, the nodes ④ and ⑥ are free nodes, which can vary their position during the optimization process. An example of the truss structure after shape and topology optimization is provided in Fig. 25.

### 7.2 22-Bar truss topology and shape design problem

The classic 18-bar cantilever truss has also been described as a problem that can be used for topology and shape optimization (Rajeev and Krishnamoorthy 1997). In this section, we have extended the 18-bar truss optimization

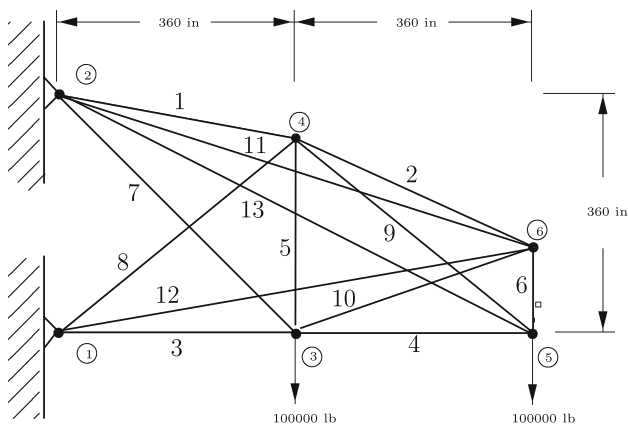


Fig. 24 The 13-bar truss with all bars present

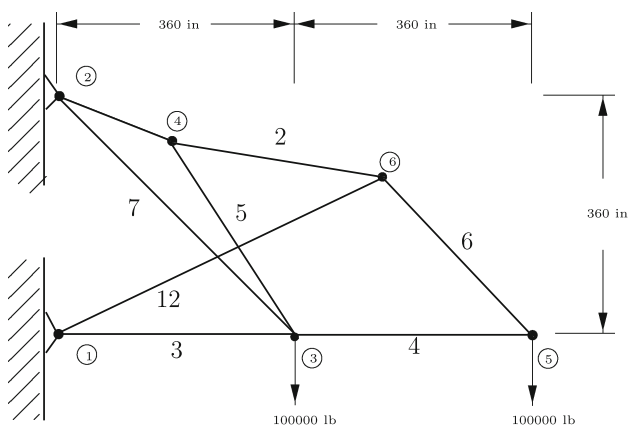


Fig. 25 An example of the 13-bar truss structure after shape and topology optimization

problem to a 22-bar problem. Figure 26 shows the relationship between beam members. Like the classic 18-bar truss this truss has 11 joints. Nodes 10 and 11 are support joints, and 5 joints are load bearing joints. The remaining four joints are free nodes that can have their positions varied during optimization. Each of these free joints can have its position varied on two axes, resulting in eight decision variables for node positions for the nodes 3, 5, 7, and 9. The total number of decision variables is 30; 22 cross-sectional areas, and 8 position coordinates for the free nodes. An example of the 22-bar truss after topology and shape optimization is provided in Fig. 27. The range of cross-sectional areas as reported by Rajeev and Krishnamoorthy (1997) is between 2 and 20 in.<sup>2</sup>. For the purposes of this study, the critical area  $\epsilon_1$  was set to 2.0 in.<sup>2</sup>.

### 7.3 Parameters for truss problems

Tables 1 and 2 detail the setup parameters associated with each truss structure studied.

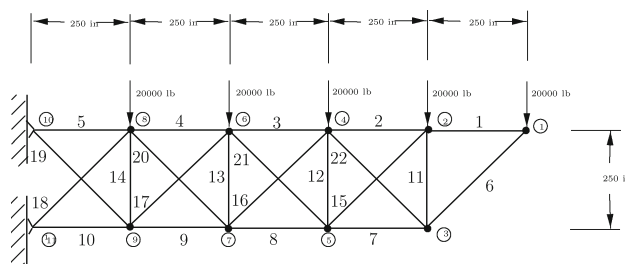


Fig. 26 The 22-bar truss with all bars present

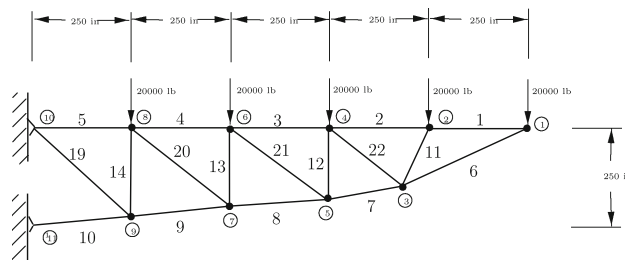


Fig. 27 An example of the 22-bar truss structure after shape and topology optimization

Table 1 13-Bar truss parameters

|   |                           |
|---|---------------------------|
| Young's modulus ( $E$ )                         | 10,000 ksi                |
| Cross-sectional area ( $A$ )                    | $\pm 35$ in. <sup>2</sup> |
| Strength (tension and compression) ( $\sigma$ ) | $\pm 25$ ksi              |
| Density ( $\rho$ )                              | 0.1 lb in. <sup>-3</sup>  |
| Critical Area ( $\epsilon_1$ )                  | 0.1 in. <sup>2</sup>      |
| Support nodes                                   | 1, 2                      |
| Load bearing nodes                              | 3, 5                      |
| y load on 3                                     | -100,000 lb               |
| x load on 3                                     | 0 lb                      |
| y load on 5                                     | -100,000 lb               |
| x load on 5                                     | 0 lb                      |
| y def. on 1                                     | 0 in.                     |
| x def. on 1                                     | 0 in.                     |
| y def. on 2                                     | 0 in.                     |
| x def. on 2                                     | 0 in.                     |
| x range of 4                                    | 0 to 720 in.              |
| y range of 4                                    | 0 to 360 in.              |
| x range of 6                                    | 0 to 720 in.              |
| y range of 6                                    | 0 to 360 in.              |

## 8 Performance in the optimization of truss structure and topology

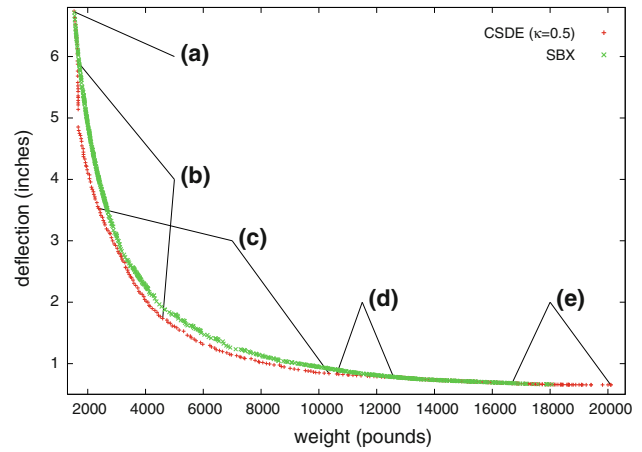
In this section the resulting performance of each of the algorithm variants is reported on the two truss problems studied. In the 13-bar problem, there are 17 decision parameters requiring optimization. This problem

**Table 2** 22-bar truss parameters

|   |                                     |
|---|-------------------------------------|
| Young's modulus ( $E$ )                       | 10,000 ksi                          |
| Cross-sectional Area ( $A$ )                  | $\pm 20 \text{ in.}^2$              |
| Stress (tension and compression) ( $\sigma$ ) | $\pm 25 \text{ ksi}$                |
| Density ( $\rho$ )                            | $0.1 \text{ lb in.}^{-3}$           |
| Critical Area ( $\epsilon_1$ )                | $2.0 \text{ in.}^2$                 |
| Support nodes                                 | ⑩, ⑪                                |
| Load bearing nodes                            | ①, ②, ④, ⑥, ⑧                       |
| y load on ①                                   | $-20,000 \text{ lb}$                |
| x load on ①                                   | $0 \text{ lb}$                      |
| y load on ②                                   | $-20,000 \text{ lb}$                |
| x load on ②                                   | $0 \text{ lb}$                      |
| y load on ③                                   | $0 \text{ lb}$                      |
| x load on ③                                   | $0 \text{ lb}$                      |
| y load on ④                                   | $-20,000 \text{ lb}$                |
| x load on ④                                   | $0 \text{ lb}$                      |
| y load on ⑤                                   | $0 \text{ lb}$                      |
| x load on ⑤                                   | $0 \text{ lb}$                      |
| y load on ⑥                                   | $-20,000 \text{ lb}$                |
| x load on ⑥                                   | $0 \text{ lb}$                      |
| y load on ⑦                                   | $-20,000 \text{ lb}$                |
| x load on ⑦                                   | $0 \text{ lb}$                      |
| y load on ⑧                                   | $-20,000 \text{ lb}$                |
| x load on ⑧                                   | $0 \text{ lb}$                      |
| y load on ⑨                                   | $0 \text{ lb}$                      |
| x load on ⑨                                   | $0 \text{ lb}$                      |
| y def. on ⑩                                   | $0 \text{ in.}$                     |
| x def. on ⑩                                   | $0 \text{ in.}$                     |
| y def. on ⑪                                   | $0 \text{ in.}$                     |
| x def. on ⑪                                   | $0 \text{ in.}$                     |
| x range of ③                                  | $750 \text{ to } 1250 \text{ in.}$  |
| y range of ③                                  | $0 \text{ to } 250 \text{ in.}$     |
| x range of ⑤                                  | $500 \text{ to } 1,000 \text{ in.}$ |
| y range of ⑤                                  | $0 \text{ to } 250 \text{ in.}$     |
| x range of ⑦                                  | $250 \text{ to } 750 \text{ in.}$   |
| y range of ⑦                                  | $0 \text{ to } 250 \text{ in.}$     |
| x range of ⑧                                  | $0 \text{ to } 500 \text{ in.}$     |
| y range of ⑧                                  | $0 \text{ to } 250 \text{ in.}$     |

demonstrates the utility of emphasizing correlated directional vectors in CSDE ( $\kappa = 0.5$ ).

In Fig. 28, the non-dominated solutions resulting from all 50 runs are plotted for the SBX and CSDE ( $\kappa = 0.5$ ) algorithms. The first and last locations at which a unique truss structure occurred from the CSDE algorithm runs are labeled. In addition, we can see in this figure that the range specified by (B) overlaps with the range specified by (C). In this region, solutions may have the truss structure of either (B) or (C), as specified in Table 3. In Table 3, the weight and deflection values are provided for the labeled solutions



**Fig. 28** Non-dominated solutions over all 50 runs at generation 50 for SBX and CSDE ( $\kappa = 0.5$ ) on the 13-bar truss problem. (A), (B), (C), (D), and (E), respectively, define the location or range over which a unique truss structure occurs

in Fig. 28, along with the actual truss structures. Furthermore, the nearest solution to the labeled CSDE solution, which resulted from the SBX variant, is also provided for comparative purposes.

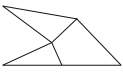
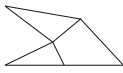
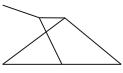
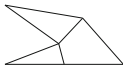
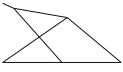
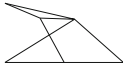
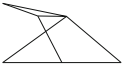
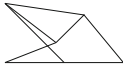
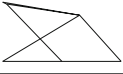
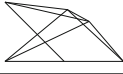
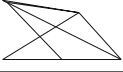
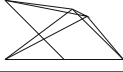
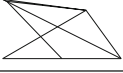
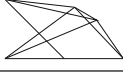
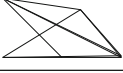
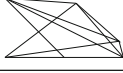


From Fig. 28 and the corresponding Table 3, it is apparent that solution (A) resulting from CSDE ( $\kappa = 0.5$ ) is lighter than the nearest solution provided by SBX, but has marginally worse deflection. In addition, the truss structure and topology is the same as the solution discovered by SBX. In all respects, the two solutions are comparable.

Solutions over the range of (B) are significantly different, in that the CSDE ( $\kappa = 0.5$ ) was able to discover a unique structure and topology with one less member than the solutions discovered with SBX. This resulted in a better truss structure with less deflection and lower weight.

Furthermore, the solutions discovered over the range of (C) demonstrate that better or comparable solutions were generated by the CSDE approach, compared with SBX. Most of the solutions over this range had lower weight and deflection than the nearest solutions discovered by SBX. The higher weighing solutions from CSDE over this range were slightly heavier than the nearest solutions from SBX, but they also exhibited a lower deflection. Equally important, it seems that the CSDE approach began to discover solutions that double up one of the truss members to strengthen the structure.

The trend of doubling up one of the truss members continued over the range of (D), where an additional member was added and the weight of the truss increased as a result. Although the solutions discovered by CSDE were heavier, the structures that were discovered exhibited less deflection than the nearest corresponding solutions from

**Table 3** Structures discovered by SBX and CSDE ( $\kappa = 0.5$ ) on the 13-bar truss problem. (A), (B), (C), (D), and (E) correspond to the labels in Fig. 28

| label | CSDE $\kappa = 0.5$ |                  |   | SBX          |                  |  |
|-------|---------------------|------------------|---|--------------|------------------|--|
|       | weight (lb.)        | deflection (in.) | structure   | weight (lb.) | deflection (in.) | structure  |
| (A)   | 1527.59             | 6.73             |    | 1534.79      | 6.72             |    |
| (B)   | 1646.39             | 5.92             |    | 1689.84      | 5.94             |    |
|       | 4595.24             | 1.73             |    | 4670.77      | 1.88             |    |
| (C)   | 2373.88             | 3.53             |    | 2655.05      | 3.54             |    |
|       | 10349.76            | 0.84             |    | 10330.61     | 0.92             |    |
| (D)   | 10616.32            | 0.83             |    | 10603.35     | 0.90             |    |
|       | 12583.42            | 0.78             |    | 12576.87     | 0.78             |    |
| (E)   | 16682.72            | 0.68             |  | 16702.17     | 0.68             |  |
|       | 20103.84            | 0.65             |  | 18065.79     | 0.66             |  |

SBX. In addition, the structures discovered by CSDE over the range of (D) were simpler and had two less members.

In the region of very heavy truss structures specified by (E), it is clear that CSDE discovered the importance of doubling up members to increase the rigidity of the truss. The first of these structures and topologies discovered by CSDE was lighter than the nearest solution from SBX, with comparable deflection. At the extreme, the heaviest solution discovered by CSDE had slightly better deflection than the nearest solution from SBX, but was heavier. It is clear from Table 3 that SBX attempted to discover solutions which are similar to the solutions discovered by CSDE, but the members did not overlap to the same degree as the solutions that CSDE discovered.

### 8.1 22-Bar truss topology and shape design problem

In the 22-bar problem there are more free nodes, and more cross-sectional areas to optimize, resulting in a total number of 30 decision variables. On this problem, the

results are reported at generation 200, as it requires more evaluations across all algorithm variants to find a reasonable number of feasible solutions.

It is important to note that on this problem, it is harder to find feasible solutions early in the search and in this particular type of problem finding feasible kinematically stable solutions which are also light structures is very difficult. This was also the motivation for using a larger population size on this problem.

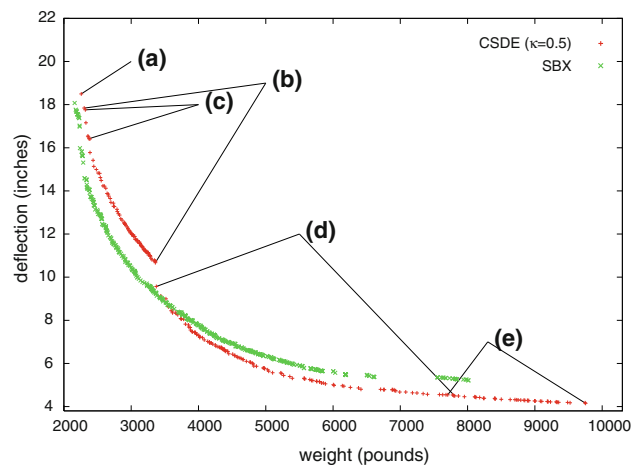
In Table 4, the weights, deflections and structures of CSDE ( $\kappa = 0.5$ ) are presented, along with the corresponding solutions from SBX. Figure 29 indicates how the labeled solutions correspond with points on the non-dominated front. The solutions that are presented are the non-dominated solutions out of all 50 runs. These results indicate that the regions labeled by (A), (B), and (C) resulting from the CSDE algorithm are dominated by solutions generated by SBX. In these regions, SBX was able to discover structures with lower weight and deflection.

**Table 4** Structures discovered by SBX and CSDE ( $\kappa = 0.5$ ) on the 22-bar truss problem

| label | CSDE $\kappa = 0.5$ |                  |           | SBX          |                  |           |
|-------|---------------------|------------------|-----------|--------------|------------------|-----------|
|       | weight (lb.)        | deflection (in.) | structure | weight (lb.) | deflection (in.) | structure |
| (A)   | 2257.67             | 18.50            |           | 2159.45      | 18.07            |           |
| (B)   | 2301.72             | 17.84            |           | 2175.26      | 17.76            |           |
|       | 3359.59             | 10.67            |           | 3266.83      | 9.59             |           |
| (C)   | 2318.13             | 17.76            |           | 2214.31      | 17.52            |           |
|       | 2373.94             | 16.42            |           | 2238.37      | 15.98            |           |
| (D)   | 3373.85             | 9.57             |           | 3316.65      | 9.40             |           |
|       | 7807.71             | 4.50             |           | 7813.47      | 5.28             |           |
| (E)   | 7705.09             | 4.54             |           | 7731.48      | 5.32             |           |
|       | 9753.78             | 4.15             |           | 8011.24      | 5.21             |           |

(A), (B), (C), (D), and (E) correspond to the labels in Fig. 29

In the region specified by (D), where CSDE was able to discover a set of unique solutions, most of the solutions corresponding to the topology and structure detailed in Table 4 had lower weight and deflection than the corresponding nearest solutions discovered by SBX. Furthermore, it is apparent from Fig. 29 that in region (D) and (E) the SBX variant could not find many solutions. CSDE was capable of finding unique structures and topologies in region (E) with lower deflection. Moreover, the topologies discovered in the region indicated by (E) demonstrate that CSDE was able to evolve solutions that double-up a beam member which makes the structure more rigid. From Fig. 29, it is clear that the regions indicated by (D) and (E) are a significant portion of the non-dominated front, clearly demonstrating that NSGA-II using CSDE ( $\kappa = 0.5$ ) is an attractive alternative to NSGA-II using SBX on this problem because it can find many better solutions relatively early in the search.



**Fig. 29** Non-dominated solutions over all 50 runs at generation 200 for SBX and CSDE ( $\kappa = 0.5$ ) on the 22-bar truss problem. (A), (B), (C), (D), and (E), respectively, define the location or range over which a unique truss structure occurs

The good performance of CSDE ( $\kappa = 0.5$ ) in the tail end of the non-dominated set, which was also noted on the 10 and 13-bar truss problems, can be accounted for when the Hessian matrix is considered. In the tail end of the search space, the second-order partial derivative values have a larger magnitude, indicating significant parameter interactions. Parameter interactions are far more prevalent in the tail end in comparison to other regions of the non-dominated front where the second-order partial derivatives have smaller values. The advantage that NSGA-II with SBX apparently has where it outperforms CSDE ( $\kappa = 0.5$ ) is because in this region of the search space there is greater separability between decision space parameters. Similarly, CSDE ( $\kappa = 0.5$ ) outperforms NSGA-II with SBX in region (D) and (E) because there are more parameter interactions there.

On the 22-bar problem, it found many solutions with significantly lower deflection because it can deal with parameter interactions more effectively in this region of the front. In contrast, NSGA-II with SBX could only find a few because of its poorer performance on problems with parameter interactions. This is of importance, because in practice one would probably not be interested in the solutions with high deflections found by NSGA-II with SBX. Interestingly, CSDE had difficulty finding low weight feasible structures for the 22-bar truss problem. In this region of the search space it is very difficult to find feasible solutions, and it's possible that the vector-wise sampling of CSDE in the infeasible space does not provide for sufficient diversity because of the discretized constraint violation values that are used as a measure of 'fitness' to guide the search. On the 13-bar problem many of the non-dominated solutions discovered by CSDE ( $\kappa = 0.5$ ) over 50 runs were better than the nearest solutions discovered by NSGA-II with SBX. The diversity that is provided through the CSDE sampling scheme is important because it enables DE to find feasible solutions faster than a DE scheme that does not incorporate such sampling. A reproduction operator such as CSDE ( $\kappa = 0.5$ ), that samples the space using directional information, is more efficient in this regard. We can also see that CSDE ( $\kappa = 0.5$ ) demonstrates that directional information combined with sampling, is critical to improving the performance of differential evolution on the truss optimization problems investigated here.

## 9 Implications and conclusion

In this work, we have addressed the stagnation issue discussed by Lampinen and Zelinka (2000) with the CSDE approach. Until now, to overcome stagnation in DE a very large population size had to be employed, or crossover was used to add more sampling diversity even though crossover

is typically ineffective when optimization problems have many parameter interactions. Furthermore, rotationally invariant DE applied to non-separable problems is limited to rather low decision space dimensions and is highly dependent on population size. In contrast, the CSDE approach is insensitive to population size on the test problems used, even though it does not employ crossover in the traditional sense. It can also handle problems with parameter interactions in high dimensional spaces very well even though it is not a strict rotationally invariant algorithm.

CSDE ( $\kappa = 0.5$ ), which incorporates directional information and also emphasizes sampling around the 'better' point as well as performing rotationally invariant correlated sampling, demonstrated rapid convergence with respect to the hyper-volume indicator on the multi-objective test problems. This indicates that it may be usefully applied to problem domains where the evaluation function is expensive to evaluate because of its rapid convergence characteristics. Furthermore, it is apparent that CSDE is highly scalable in the decision space in multi-objective problem domains as well, even to large numbers of decision variables, thanks to the diversity provided by its sampling scheme in combination with its ability to direct the search towards more optimal regions. The CSDE approach addresses the issue highlighted by Sutton et al. (2007) where high selection pressure can cause a rapid loss of diversity and therefore a decrease in exploration potential. This was demonstrated on multi-modal Problem P4 in a decision space of 200 dimensions, with a population size of only 100 individuals.

In this paper, we also addressed the question of whether or not the proposed CSDE algorithm is efficient and competitive in comparison with a multi-objective genetic algorithm, namely NSGA-II with SBX. A comparison of the directional information variants CSDE ( $\kappa = 0$ ) and ( $\kappa = 0.5$ ), within the NSGA-II framework, was performed on two plane truss multi-objective optimization problems. The major results from this are that the CSDE ( $\kappa = 0.5$ ) variant that uses directional information with an equal emphasis on C-sampling and UC-sampling demonstrated superior or highly competitive performance against SBX. In addition, CSDE ( $\kappa = 0.5$ ) was capable of finding many better truss structures that NSGA-II with SBX could not find over 50 runs.

The results presented in this paper are significantly important to practitioners who are interested in optimizing non-separable problems. Until now, previous work in this area focussed on computationally expensive Evolutionary Strategy techniques. We have presented a computationally efficient, simple optimization algorithm for dramatically improving optimization performance on non-separable problems in high dimensional spaces.

In future work, we intend to investigate the comparative performance of the CSDE approach on other state-of-the-art DE algorithms and Evolution Strategies intended for parameter-interaction problems, in addition to investigating the scalability of the CSDE on much larger truss topology and design problems.

## References

- Coello CAC, Christiansen AD (2000) Multiobjective optimization of trusses using genetic algorithms. *Comput Struct* 75:647–660
- Deb K, Gulati S, Chakrabarti S (1998) Optimal truss-structure design using real-coded genetic algorithms. In: *Genetic programming 1998: Proceedings of the third annual conference*, University of Wisconsin, Madison, Wisconsin, USA. Morgan Kaufmann, Massachusetts, pp 479–486
- Deb K, Khan N, Jindal S (2000) Optimal truss-structure design for multiple objectives. In: *Tenth national Seminar on aerospace structures*, Kanpur, India. Indian Institute of Technology, pp 168–180
- Deb K, Agrawal S, Pratap A, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput* 6(2):182–197
- Deb K, Sinha A, Kukkonen S (2006) Multi-objective test problems, linkages, and evolutionary methodologies. In: *GECCO 2006 Genetic and evolutionary computation conference*, Seattle, Washington, USA (July 8–12 2006). ACM Press, New York, pp 1141–1148
- Espi MV (1998) Genetic algorithms-based methodologies for design optimization of trusses (discussion). *J Struct Eng* 979–980
- Fleischer M (2003) The measure of pareto optima: applications to multiobjective metaheuristics. In: *Proceedings of evolutionary multicriterion optimization: Second International Conference, EMO 2003* (April 8–11 2003). Lecture Notes in Computer Science. Springer, Faro, vol 2632
- Hibbeler RC (2006) 14. In: *Structural analysis*. Pearson Prentice Hall, Upper Saddle River, pp 529–564
- Ilonen J, Kamarainen JK, Lampinen J (2003) Differential evolution training algorithm for feed-forward neural networks. *Neural Process Lett* 7(1):93–105
- Iorio A, Li X (2006) Rotated test problems for assessing the performance of multi-objective optimization algorithms. In: *GECCO 2006 genetic and evolutionary computation conference*, Seattle, Washington, USA (July 8–12 2006). ACM Press, New York, pp 683–690
- Iorio A, Li X (2008) Rotated problems and rotationally invariant crossover in evolutionary multi-objective optimization. *Int J Comput Intell Appl Spec Issue Simul Evol Learn* 7(2):149–186
- Lampinen J, Zelinka I (2000) On stagnation of the differential evolution algorithm. In: *Proceedings of MENDEL 2000, 6th international Mendel conference on soft computing*, pp 76–83
- Li X, Iorio AW (2008) Improving the performance and scalability of differential evolution. In: *Simulated evolution and learning: 6th international conference, SEAL 2008*, Melbourne, Australia (December 7–10 2008). Lecture Notes in Computer Science. Springer, Berlin, vol 5361, p 131140
- Newham LTH, Nortonb JP, Prosserd IP, Crokee BFW, Jakemane AJ (2003) Sensitivity analysis for assessing the behaviour of a landscape-based sediment source and transport model. *Environ Model Softw* 18(8–9):741–751
- Pelikan M, Goldberg DE, Cant-Paz E (2000) Linkage problem, distribution estimation, and bayesian networks. *Evol Comput* 8(3):311–340
- Price K (1996) Differential evolution: a fast and simple numerical optimizer. In: *Biennial conference of the North American Fuzzy Information Processing Society—NAFIPS*, Berkeley, California, USA (June 19–22, 1996). IEEE, vol 3339, pp 524–527
- Price K (1999) An introduction to differential evolution. In: *New ideas in optimization*. McGraw Hill, New York
- Rajeev S, Krishnamoorthy CS (1997) Genetic algorithms-based methodologies for design optimization of trusses. *J Struct Eng* 123:350–358
- Reeves C, Wright C (1995) An experimental design perspective on genetic algorithms. In: *Foundations of genetic algorithms*. Morgan Kaufmann, Massachusetts, vol 3, pp 7–22
- Rogalsky T, Derksen RW, Kocabiyik S (1999) Differential evolution in aerodynamic optimization. In: *Proceedings of the 46th annual conference of the Canadian Aeronautics and Space Institute*, Montreal, Quebec (May 3–5, 1999), pp 29–36
- Ruy WS, Yang YS (2001) Topology design of truss structures in a multicriteria environment. *Computer Aided Civil Infrastruct Eng* 16:246–258
- Salomon R (1996) Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions: a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems* 39(3):263–278
- Salomon R (1997) The evolution of different neuronal control structures for autonomous agents. *Robot Autonom Syst* 22(3–4): 119–213
- Storn R (1996) Differential evolution design of an IIR-filter. In: *Proceedings of IEEE international conference on evolutionary computation, ICEC '96*, Nayoya University, Japan, IEEE (May 20–22 1996), pp 268–273
- Sutton AM, Lunacek M, Whitley LD (2007) Differential evolution and non-separability: using selective pressure to focus search. In: *GECCO 2007 genetic and evolutionary computation conference*. ACM Press, London, pp 1428–1435
- Whitley LD (1991) Fundamental principles of deception in genetic search. In: *Foundations of genetic algorithms*. Morgan Kaufmann, Massachusetts, pp 221–241
- Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms—a comparative case study. In: *Parallel problem solving from nature—PPSN V: 5th international conference, Proceedings* (September 27–30 1998). Lecture Notes in Computer Science. Springer, Amsterdam, vol 1498, pp 292–304