A NEW MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM: NEIGHBORHOOD EXPLORING EVOLUTION STRATEGY

Xiaolin Hu
Automotive Engineering Institute
Wuhan University of Technology
Wuhan 430070, China
x.l.hu@l63.com

Carlos A. Coello Coello
CINVESTAV-IPN
Av. IPN No. 2508
Col. San Pedro Zacatenco
México, D.F. 07300, México
ccoello@cs.cinvestav.mx

Zhangcan Huang
School of Science
Wuhan University of Technology
Wuhan 430070, China
huangzc@mail.whut.edu.cn

This paper proposes a new multi-objective evolutionary algorithm, called *neighborhood exploring evolution strategy* (NEES). This approach incorporates the idea of neighborhood exploration together with other techniques commonly used in the multi-objective evolutionary optimization literature (namely, non-dominated sorting and diversity preservation mechanisms). This idea of the proposed approach was derived from a single-objective evolutionary algorithm, called *line-up competition algorithm* (LCA). The main idea is to assign neighborhoods of different size to different solutions. Within each neighborhood, new solutions are generated using a (1+λ)-ES (evolution strategy). This scheme naturally balances the effect of local search (which is done by the evolution strategy) with that of the global search performed by the algorithm, and gradually impels the population to progress towards the true Pareto-optimal front of the problem and to explore the extent of such front. Three versions of our proposal are studied: a (1+1)-NEES, a (1+2)-NEES and a (1+5)-NEES. Such approaches are validated on a set of standard test problems reported in the specialized literature. Simulation results indicate that, for continuous numerical optimization problems, our proposal (particularly the (1+1)-NEES) is competitive with respect to the NSGA-II, which is an algorithm representative of the state-of-the-art in evolutionary multi-objective optimization. Moreover, all the versions of our NEES improve on the results of the NSGA-II when dealing with a discrete optimization problem. Although preliminary, such results might indicate a potential application area in which our proposed approach could be particularly useful.

Keywords: Neighborhood exploring evolution strategy; Line-up competition algorithm; multi-objective optimization; evolutionary algorithm; evolution strategy

1 INTRODUCTION

Multi-objective optimization problems arise in a natural way in many real-world applications and their importance has considerably increased in the last few years. Because of their nature, multi-objective optimization problems tend to present several solutions (all of which are equivalent among themselves) and therefore require different optimization algorithms than those traditionally used for global (single-objective) optimization.

Over the years, a significant number of mathematical programming techniques have been developed to solve multi-objective optimization problems [1]. However, such techniques have several limitations. For example, some techniques are only applicable to convex Pareto fronts. Others require that the objective functions (and perhaps also the constraints) of the problem are differentiable. In general, they all require an initial guess of the location of the Pareto front and they produce a single non-dominated solution per run.

As an alternative to the use of mathematical programming techniques, during the last few years, a number of heuristics have been proposed to solve multi-objective optimization problems [2-6]. From them, we will focus specifically on evolutionary algorithms, which are a heuristic inspired on natural selection. The use of evolutionary algorithms for multi-objective optimization presents several advantages. For example, these algorithms are less susceptible to the shape or continuity of the Pareto front, they do not require an initial (guessed) solution from the user, they are population-based (i.e., they operate simultaneously with several solutions at each iteration) and, in consequence, they can generate several non-dominated solutions in one run [7].

The first multi-objective evolutionary algorithm (MOEA) is the *vector evaluated genetic algorithm* (VEGA), which dates back to the mid-1980s [8]. Since then, a wide number of MOEAs have been proposed in the specialized literature (see [7, 9] for a comprehensive overview of MOEAs).

The purpose of this paper is to extend a single-objective optimization algorithm, called *line-up competition algorithm* (LCA) so that it can deal with multi-objective optimization problems. LCA is a population-based global search algorithm, originally proposed by Yan in his dissertation [10]. LCA has been successfully applied to continuous numerical optimization problems and many combinatorial optimization problems such as the 0/1 knapsack problem and the traveling salesperson problem [11-13]. To the authors' best knowledge, the work

reported in this paper constitutes the first attempt to extend LCA for solving multi-objective optimization problems. Additionally, we also analyze the possible advantages and disadvantages of our proposed approach when dealing with both continuous and discrete multi-objective optimization problems.

2 BASIC CONCEPTS

First, we will introduce some basic definitions related to multi-objective optimization that are required to make the paper self-contained.

Definition 1 (Multi-Objective Optimization Problem) A multi-objective optimization problem can be defined as:

$$\left.\begin{array}{lll} \min & f_m(\mathbf{x}), & m = 1,2,\cdots,M; \\ \text{s.t.} & g_j(\mathbf{x}) \geq 0, & j = 1,2,\cdots,J; \\ & h_k(\mathbf{x}) = 0, & k = 1,2,\cdots,K; \\ & x_i \in [x_i^L, x_i^U], & i = 1,2,\cdots,n \end{array}\right\} \quad (1)$$

In this paper, we only consider unconstrained multi-objective optimization problems, and thus the $g_j(\mathbf{x})$ and $h_k(\mathbf{x})$ functions are not considered in our case. Without loss of generality all the objectives are assumed to be minimized.

Definition 2 (Pareto dominance) Solution $\mathbf{x}^{(1)}$ is said to *dominate* (in a Pareto sense) another solution $\mathbf{x}^{(2)}$ if *both* of the following conditions are satisfied:

(1) $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in all the objectives, or $f_j(\mathbf{x}^{(1)}) \leq f_j(\mathbf{x}^{(2)})$ *for all* $j = 1, 2, \ldots, M$.

(2) $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in at least one objective, or $f_j(\mathbf{x}^{(1)}) < f_j(\mathbf{x}^{(2)})$ for at least one j $\{j=1, 2, \ldots, M\}$.

Definition 3 (Pareto-optimal set and Pareto front) Among a set of solutions P, the non-dominated set of solutions P' are those that are not dominated by any other member of the set P.

The set P' is also called *Pareto-optimal set.* The objective function values corresponding to the decision variables contained in P' constitute the so-called *Pareto front*.

The two main goals in multi-objective optimization are [7, 9]:

- To find a set of solutions as close as possible to the true Pareto-optimal set.
- To find a set of solutions as diverse as possible.

3 THE LINE-UP COMPETITION ALGORITHM FOR SINGLE-OBJECTIVE OPTIMIZATION

The LCA is a type of evolutionary algorithm with some important differences. Its basic structure is described in Table I.

TABLE I The Main Loop of the Line-up Competition Algorithm

Uniformly generate the initial population of size N in the entire search space, and evaluate the population
Repeat
Sort the N parents in an ascending sequence (called a *line-up*) according to their objective values (we are assuming minimization problems)
Assign each parent a neighborhood according to its position in the line-up, satisfying that the sizes of the assigned neighborhoods from the first to the last parent are in an ascending sequence. So, the first parent gets the smallest neighborhood while the last parent gets the largest neighborhood
Each parent mutates λ times and produces λ offspring within its assigned neighborhood. The parent and its offspring then constitute a family (there exists a total of N families so far)
The (λ +1) individuals in every family compete with each other, and the best one survives as a parent for the next generation
Contract the neighborhoods that will be assigned to the N parents
Until the terminal condition is met

In the LCA, at the first stage, the parents are sorted in a sequence (called a *line-up*) to compete with each other. At the second stage, several families evolve independently and each family chooses its best individual as the parent for the next generation. The two stages are continuously executed until some terminal condition is met. There are two competition levels within the algorithm: one is the ranking competition among the families and the other is the survival competition inside a family. That is origin of the algorithm's name. As a matter of fact, some features of the LCA that its author argues as novel have been previously used in the literature. The use of several

independent families is similar to the multiple population based evolutionary algorithms, and the mechanism of generating new offspring together with the competition inside a family constitutes the principle of the so-called $(1+\lambda)$-ES (evolution strategy). However, the conception of assigning different search spaces or neighborhoods to individuals according to their ranks in a line-up is a truly novel and useful feature since it naturally balances the local search and global search performed by the algorithm. Better parents in the line-up get smaller neighborhoods, which results in a fast convergence towards the local optima; and worse parents get larger neighborhoods, which promotes the global search. Another novel feature of the LCA is its adaptive neighborhood contraction. This technique accelerates the convergence rate and improves the quality of the approximation obtained. The idea as well as its implementation is comparatively simple, which can be regarded as another merit of the algorithm. Additionally, this approach has been found to be quite effective in several problems that span numerical and combinatorial optimization as well as some real-world applications [10-13].

4 A NAIVE NEIGHBORHOOD EXPLORING EVOLUTION STRATEGY

4.1 Non-dominated Sorting

One critical step in the process of the LCA is to sort the population according to the individuals' fitness at every generation. This task can be easily carried out since for single-objective problems the relationship between any pair of solutions is obvious. However, for multi-objective problems, determining which solution is better and which is worse is not completely straightforward (e.g., two solutions may be incomparable). In order to sort the population like in the original LCA, an intuitive idea is to use the *Pareto dominance* concept defined in Section 2. If solution A dominates solution B, it means that solution A is *better* than B. However, not every pair of solutions can be differentiated using the *Pareto dominance* concept because it is possible that two given solutions are non-dominated with respect to each other (i.e., in this case, none of them is better than the other).

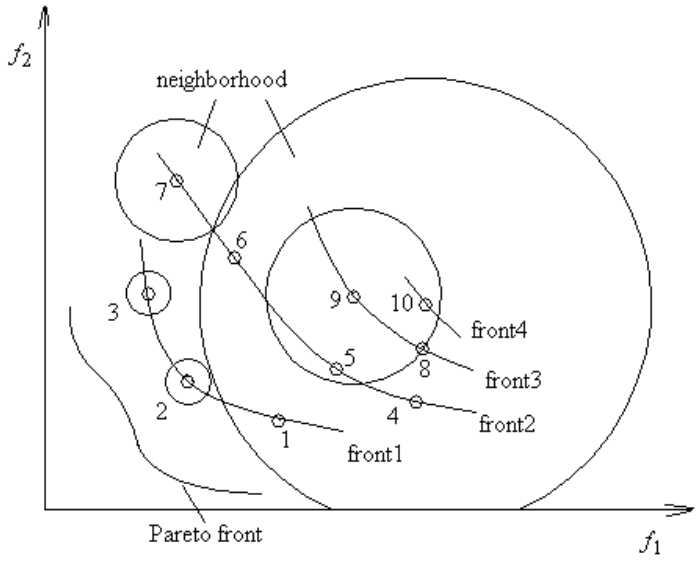

FIGURE 1 Illustration of the non-dominated fronts in the objective space for a two-objective optimization problem

We propose to extend the idea of the line-up in the LCA such that instead of restricting one position in a line-up to be able to hold only one individual, we allow that one position can hold multiple solutions. This will be more suitable for dealing with multi-objective optimization problems. By adopting the Pareto dominance concept, any population can be classified into different non-dominance levels. The classification can be easily carried out by gradually disregarding the non-dominated fronts previously found and identifying the new front of the resulting set of solutions. The fast non-dominated sorting approach proposed by Deb et al. [14] can be adopted for this sake.

Figure 1 illustrates an example of such classification for a two-objective problem. There are ten solutions in objective function space that are classified into four groups, {1,2,3}, {4,5,6,7}, {8,9}, {10}, which are numbered as fronts 1 to 4. It can be seen from Figure 1 that the solutions residing in the same front are non-dominated among themselves. However, the solutions of front 2 are at least dominated by one solution of front 1; the solutions of front 3 are at least dominated by one solution of front 2; and so on. Hence, intuitively, the solutions of front 1 are *better* than the solutions of front 2; the solutions of front 2 are *better* than solutions of front 3, and so on. Then, the four fronts may constitute a *line-up*. But this line-up differs from that of the original LCA, since in this case, there are only four positions in the line-up but ten solutions are to be accommodated. Thus one position must accommodate more than one solution.

For convenience, we assign each solution a rank, that is, the number of the front to which it belongs. For instance, in Figure 1, solutions 1, 2 and 3 have rank 1, solutions 4, 5, 6 and 7 have rank 2, solutions 8 and 9 have rank 3, and solution 10 has rank 4.

It is worth indicating that the non-dominated sorting procedure adopted in this paper has a complexity $O(kGM^2)$, where k is the number of objective functions, M is the population size and G is the total number of generations (see [14]). This computational cost obviously degrades as the number of objective functions increases. Although it is possible to produce ranking algorithms with a lower computational complexity (see for example [22]), most current multi-objective evolutionary algorithms have the same (or an even higher) computational cost. An exception are the approaches based on aggregating functions, but most of them are severely limited by the shape of the Pareto front (e.g., linear aggregating functions are unable to generate concave Pareto fronts [7,9]). Therefore our decision of using non-dominated sorting instead of an aggregating function.

4.2 Neighborhood Assignment

After the sorting procedure, the next step is to decide the neighborhood for every solution in the population. As in the original LCA, we relate the size of the search space to the position of the line-up. So, the solutions with lower ranks have smaller sized neighborhoods and the solutions with the same rank have the same sized neighborhoods. Figure 1 illustrates this. Let a circle denote a neighborhood[1]. Several neighborhoods of different sizes are illustrated in Figure 1. It can be seen that solution 3 has the smallest neighborhoods, followed by solutions 7 and 9; and solution 10 has the largest neighborhood. But solution 2 and solution 3 have neighborhoods of the same size. A linear neighborhood size assignment method is adopted in this paper. Assume that there are K non-dominated fronts of different levels at the g-th generation. Let $\boldsymbol{D}_K(g)$ denote the neighborhood size assigned to the last front of solutions at the g-th generation. Then, the neighborhood size for the k-th front of solutions at this generation is

$$\boldsymbol{D}_k(g) = {}^{k}\!/_{K}\ \boldsymbol{D}_K(g) \quad (k=1, 2, \ldots, K) \tag{2}$$

For real numerical optimization, $\boldsymbol{D}_K(g)$ is a vector in R^n and a function of the generation counter g. All the neighborhoods shrink by defining a decreasing function $\boldsymbol{D}_K(g)$. The LCA suggests a simple method for this purpose and can be adopted in our proposal:

[1] Of course the circle does not indicate the actual shape of the neighborhood in objective function space. In addition, the neighborhood is defined in decision variable space, not in objective function space.

$$\boldsymbol{D}_K(g) = \eta\ \boldsymbol{D}_K(g\text{-}1) \text{ or } \boldsymbol{D}_K(g) = \eta^{g-1}\ \boldsymbol{D}_K(0)$$

where $0<\eta<1$, $g\geq 1$. The initial maximum neighborhood size in a population is $\boldsymbol{D}_K(0)=\alpha\ [\boldsymbol{U}\ \text{-}\ \boldsymbol{L}]$, where $\boldsymbol{L}$ and $\boldsymbol{U}$ represent the lower and upper bounds of the variables and α is a scaling factor in the range (0,1]. Then, Equation (2) can be expressed as

$$\boldsymbol{D}_k(g) = \frac{k\alpha\eta^{g-1}}{K}\ [\ \boldsymbol{U}\ \text{-}\ \boldsymbol{L}] \tag{3}$$

Definition 4 (Neighborhood in R^n) Let $\mathbf{x}\in \mathrm{R}^n$ be a solution residing in the k-th non-dominated front of the population at the g-th generation, and $\boldsymbol{D}_k(g)$ be its neighborhood size. The neighborhood of $\mathbf{x}$ is

$$\mathcal{N}(\mathbf{x})=[\mathbf{x}\text{-}\boldsymbol{D}_k(g)/2,\ \mathbf{x}+\boldsymbol{D}_k(g)/2]$$

Obviously the $\mathcal{N}(\mathbf{x})$ is a hyper-rectangular subspace of R^n.

Once the neighborhood for any individual in a population is decided, the $(1+\lambda)$-Evolution Strategy can be applied to the solutions as in the LCA. However, as some members of a family consisting of one parent and λ offspring may be incomparable in the Pareto dominance sense, the number of *winners* after the competition within the family is not necessarily equal to one. In other words, the mechanism of generating new solutions in the multi-objective optimization algorithm is actually no longer a strict $(1+\lambda)$-ES. Despite this difference, the new algorithm is still called evolution strategy, mainly for the sake of convenience.

4.3 Proposal of the Neighborhood Exploring Evolution Strategy

Our proposed approach to solve multi-objective optimization problems is summarized in Table II. Since the use of the neighborhood concept and the adoption of an evolution strategy are two critical characteristics that distinguish the approach from other existing multi-objective optimization algorithms, we named our proposal the *neighborhood exploring evolution strategy* (NEES).

TABLE II The Naive Neighborhood Exploring Evolution Strategy

Initialize the parent population P_1 of size N within the entire search space, and evaluate the population
Perform the non-dominated sorting procedure on P_1
Repeat
Classify P_t ($t \geq 1$) into subsets according to the non-dominated fronts and assign each solution a rank

Assign neighborhoods to the solutions in P_t according to their ranks

Each solution in P_t generates λ offspring within its assigned neighborhood to constitute a family

The $(1+\lambda)$ individuals in every family compete with each other, and the non-dominated ones of that family survive

The survivors from all N families constitute the offspring population Q_t of size $\overline{N}$, ($N \le \overline{N} \le (1+\lambda)N$)

Perform the non-dominated sorting procedure on Q_t and reduce the size of Q_t from $\overline{N}$ to N using some crowding handling technique

Contract the neighborhoods

Let $P_{t+1}= Q_t$, $t=t+1$

Until the terminal condition is met

Since each individual of P_t produces λ offspring, the total population P_t produces λN offspring. So $\overline{N}$ is at most equal to $(1+\lambda)N$ and it is always greater than or equal to N. The problem is how to reduce the size of Q_t to a fixed size N. The application of the non-dominated sorting procedure to this population is an obvious choice. After doing that, the solutions residing in the first front should be retained, followed by the solutions of the second front, the third front, and so on. When the last allowable front is being considered, there may exist more solutions in the last front than desired. If so, some solutions of this last front must be deleted. Considering that the second main goal of multi-objective optimization is to maintain the diversity of the solutions, then the most crowded solutions of this front should be the candidates for deletion. This is the truncation procedure adopted by our approach.

There are many techniques that can be used for distributing solutions in a uniform way, such as the *sharing function model* proposed by Goldberg and Richardson [15], the *clustering method* adopted in SPEA [16], and the *adaptive grid* proposed in PAES [17]. We adopt Deb et al.'s *crowding distance* technique here, which is used in the NSGA-II [14]. The basic idea is to use the average distance d_i of two solutions on either side of a particular solution i along each of the objectives. This quantity d_i serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices. For further details about the computation of the *crowding distance*, refer to [9, 14].

4.4 Example Problem

Consider a simple bi-objective problem with only two decision variables:

$$\left.\begin{aligned} &\min\ f_1(\mathbf{x}) = x_1 \\ &\min\ f_2(\mathbf{x}) = \frac{1+x_2}{x_1} \\ &x_1 \in [0.1,1],\ x_2 \in [0,3] \end{aligned}\right\} \tag{4}$$

The Pareto-optimal solutions correspond to $0 \le x_1^* \le 1$ and $x_2^* = 0$. We use this problem to show the step-by-step procedure of the proposed algorithm.

Step 1 Generate 6 solutions randomly in the entire decision space $\{(x_1, x_2) \mid x_1 \in [0.1,1], x_2 \in [0,3]\}$ and evaluate their objectives (see Table III and Figure 2). In Figure 2 the Pareto-optimal front corresponds to the horizontal axis in the left chart and the continuous curve in the right chart.

TABLE III Six Initial Solutions for a Sample Problem

Solution	x1	x2	f1	f2
s1	0.2442	2.8316	0.2442	15.6913
s2	0.4047	0.8663	0.4047	4.6113
s3	0.9749	0.0244	0.9749	1.0507
s4	0.8730	1.8880	0.8730	3.3082
s5	0.7300	2.9335	0.7300	5.3885
s6	0.9870	2.8769	0.9870	3.9278

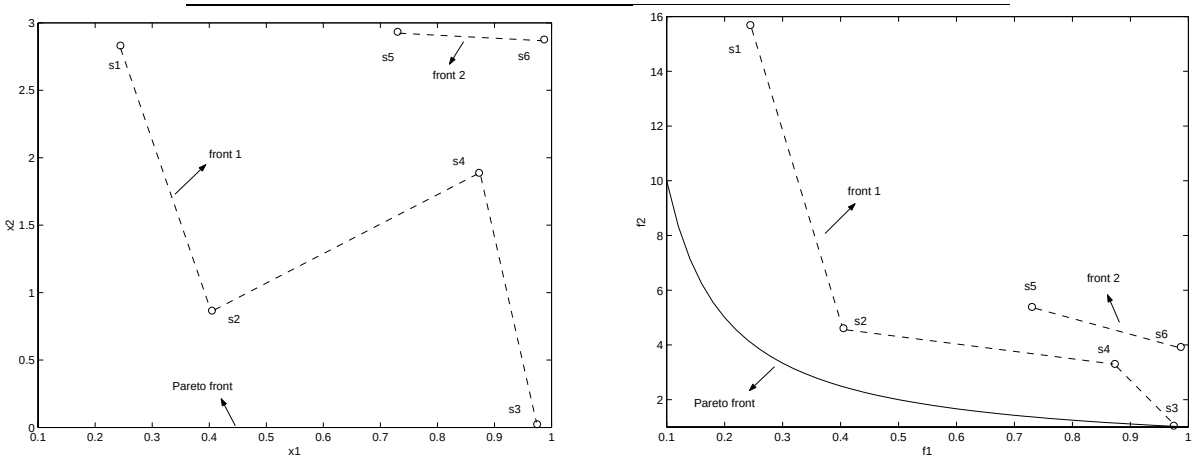

FIGURE 2 Initial solutions in decision variable space (left) and objective function space (right)

Step 2 Perform the non-dominated sorting procedure. It can be easily seen from Figure 2 that solutions {s1, s2, s3, s4} constitute the first Non-dominated front and {s4, s5} constitute the second front.

Step 3 Assign different sized neighborhoods to the six parent solutions. As there are two fronts in objective function space, the parameter *K* in Equation (3) is equal to 2. Set α to 0.4 and get

$$\boldsymbol{D}_1(1) = \frac{1 \times 0.4\eta^{1-1}}{2}[\boldsymbol{U} - \boldsymbol{L}] = 0.2[\boldsymbol{U} - \boldsymbol{L}] = 0.2\left[\begin{bmatrix}1\\3\end{bmatrix} - \begin{bmatrix}0.1\\0\end{bmatrix}\right] = \begin{bmatrix}0.18\\0.60\end{bmatrix}$$

$$\boldsymbol{D}_2(1)= \frac{2\times 0.4\eta^{1-1}}{2}[\ \boldsymbol{U}-\boldsymbol{L}] = 0.4[\boldsymbol{U}-\boldsymbol{L}]= 0.4\left[\begin{bmatrix}1\\3\end{bmatrix}-\begin{bmatrix}0.1\\0\end{bmatrix}\right]=\begin{bmatrix}0.36\\1.20\end{bmatrix}$$

Then, the neighborhoods of s1, s2, s3 and s4 are

$$\mathcal{N}(\mathbf{x}_i)=[\ \mathbf{x}_i - \frac{1}{2}\boldsymbol{D}_1(1),\ \mathbf{x}_i + \boldsymbol{D}_1(1)], \quad i=1, 2, 3, 4.$$

And the neighborhoods of s5 and s6 are

$$\mathcal{N}(\mathbf{x}_i)=[\ \mathbf{x}_i - \frac{1}{2}\boldsymbol{D}_2(1),\ \mathbf{x}_i + \boldsymbol{D}_2(1)], \quad i=5, 6.$$

TABLE IV The Parent and Offspring in Each Family

Solutions		x1	x2	f1	f2	status
family 1	s1	0.2442	2.8316	0.2442	15.6913	win
	s1'	0.2534	3.0000	0.2534	15.7824	fail
	s1''	0.2762	3.0000	0.2762	14.4846	win
family 2	s2	0.4047	0.8663	0.4047	4.6113	fail
	s2'	0.3928	1.0081	0.3928	5.1125	win
	s2''	0.3932	0.6051	0.3932	4.0823	win
family 3	s3	0.9749	0.0244	0.9749	1.0507	win
	s3'	1.0000	0	1.0000	1.0000	win
	s3''	0.9948	0	0.9948	1.0052	win
family 4	s4	0.8730	1.8880	0.8730	3.3082	win
	s4'	0.8897	1.7954	0.8897	3.1420	win
	s4''	0.9018	1.9347	0.9018	3.2544	fail
family 5	s5	0.7300	2.9335	0.7300	5.3885	win
	s5'	0.7489	2.7793	0.7489	5.0465	win
	s5''	0.6013	3.0000	0.6013	6.6527	win
family 6	s6	0.9870	2.8769	0.9870	3.9278	win
	s6'	1.0000	2.8780	1.0000	3.8780	win
	s6''	0.9141	3.0000	0.9141	4.3758	win

For the two-dimensional case, the neighborhoods are rectangles in decision variable space, as shown in Figure 3. Then, each solution generates (randomly) λ offspring in the corresponding neighborhood. Here, λ =2. The offspring are shown in Table IV and Figure 3. (Note: the neighborhoods are restricted to the feasible region, that is, if any offspring generated is outside [***L, U***], it is moved back to the corresponding bounds.)

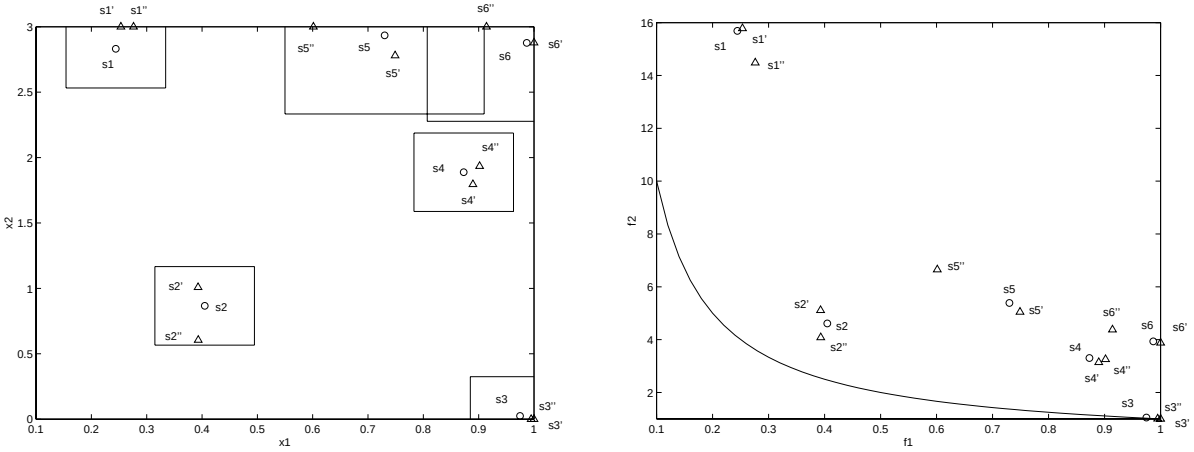

FIGURE 3 Parent solutions and offspring in decision variable space (left) and objective function space (right)

Step 4 Within each family the (1+ λ) individuals compete with each other and the winners, i.e. the non-dominated solutions, survive. In this case, most of the 18 solutions survive except for s1', s2 and s4'' that must be directly deleted. The 15 resulting solutions constitute the offspring population Q_1.

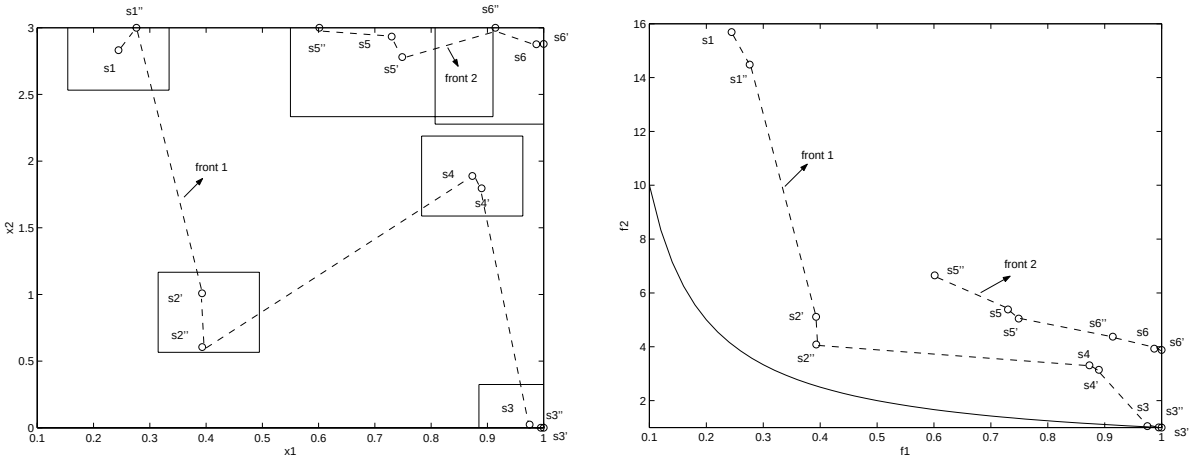

FIGURE 4 Non-dominated sorting for the set of solutions obtained

(Left: decision variable space; Right: objective function space)

Step 5 Perform the non-dominated sorting procedure on Q_1 and obtain two non-dominated fronts (see Figure 4):

Front 1= {s1, s1'', s2', s2'', s4, s4', s3, s3', s3''}

Front 2= {s5, s5', s5'', s6, s6', s6''}

Step 6 Reduce the population size from 15 to 6. As the number of solutions residing in the front 1 is already greater than 6, all the solutions residing in the front 2 must be discarded. In addition, some solutions in the front 1 residing in more crowded regions must also be discarded. By calculating Deb et al's *crowding distance* measure

we find that the solutions s3, s3’ and s4’ are the candidates for deletion (for details see [14]). The population now consists of {s1, s1’’, s2’, s2’’, s3’, s4} and becomes the parent population for the next generation.

Step 7 Shrink the neighborhoods according to Equation (3) or using other contraction technique.

Step 8 Go to **step 3** until the termination condition is satisfied.

5 IMPROVEMENTS TO THE NAIVE NEIGHBORHOOD EXPLORING EVOLUTION STRATEGY

The NEES described above is a straightforward extension of the LCA, since it directly adopts many basic ideas of the LCA. However, whether these ideas still make sense for multi-objective optimization is questionable. Intuitively, we find that at least three issues need to be re-examined: the population classification approach for neighborhood assignment, the competition conception inside the individual families and the neighborhood contraction technique.

5.1 Modified population classification approach

Consider the first idea stated above (i.e., the population classification approach). Both the LCA and the naive NEES use it to classify the population satisfying that the *best* solutions get smaller sized neighborhoods and the *worst* solutions get greater sized neighborhoods. In fact, this idea can be improved for multi-objective optimization. Assume a solution set A is said to be *better* than a solution set B (in a Pareto dominance sense). Shall we assign some of the solutions in A neighborhoods of the same size as those assigned to B? If so, what is the effect of doing this? For example, in Figure 1, we do not assign solution 1 a neighborhood of the same size as those assigned to solutions 2 and 3; instead we assign solution 1 a neighborhood as large as that of solution 7 or solution 9 or solution 10. Obviously the total search space of the current population will get closer to the Pareto front located at the bottom-left of Figure 1. Therefore, such a modification has potential to accelerate the convergence rate of the algorithm. The possible disadvantage of this approach is that there will be an obvious increase of the selection pressure that our original neighborhood assignment did not have. Such a high selection pressure may cause premature convergence. However, we argue that if the parameters of the approach are properly set, it is possible to balance this increase in the selection pressure with the exploration in the neighborhoods and the resulting algorithm turns our to be more competitive. Hence, we modified the assignment procedure based on the above observation.

The principle of the proposed method is to reclassify the population into *R pseudo* fronts with a pre-defined number of individuals in each pseudo front. For instance, a geometric distribution can be adopted for this purpose:

$$n_k = rn_{k-1},\ k=2, \ldots, R \tag{5}$$

where n_k is the number of individuals in the k-th front and r (<1) is the reduction rate. For a given population with N individuals, n_k can be calculated as

$$n_k = N\frac{1-r}{1-r^R}r^{k-1} \tag{6}$$

Since r <1, the number of individuals in the first pseudo front is highest. Thereafter, each pseudo front has an exponentially reducing number of solutions. A small r results in more solutions in the first pseudo front and hence emphasizes the local search. On the contrary, a greater r results in more solutions in the last pseudo front and hence emphasizes the global search. However, this exponential distribution is an assumption and it is desirable to attempt other choices such as an arithmetic distribution.

The procedure to select solutions from the population to fill these pseudo fronts is stated below. After the truncation procedure in the main loop of the naive NEES described in Table II, the N individuals in the population P_t is classified into the front 1, front 2, …, and front K using the non-dominated sorting approach. Then, the population is reclassified using the approach described in Table V.

TABLE V Modified Population Classification Approach

Sort the solutions in a line-up satisfying that the solutions belonging to front 1 are in front of the solutions belonging to front 2, the solutions belonging to front 2 are in front of the solutions belonging to front 3, and so on. However, there is no priority for those solutions belonging to the same front. Set i=1;
Repeat
Count n_i solutions from the beginning of the line-up to constitute the pseudo front i; (n_i is calculated using Equation (6).)
Delete the top n_i solutions from the line-up;
i= i+1;
Until no solution remains in the line-up (meanwhile i = R)

Accordingly, after this procedure, every solution has a *pseudo* rank value instead of its real rank value. The pseudo rank serves to decide the neighborhood size of a solution as the real rank does in the naive NEES. Then, the neighborhood assignment method described in Section 4.2 remains the same for the modified NEES. If we let

K denote the number of the pseudo fronts instead of the real non-dominated fronts, Equation (2) and (3) remain unchanged. For convenience, we discard the symbol *R* and use *K* instead hereafter.

It should be noted that in the naive NEES, the number of non-dominated fronts can not be controlled. As the algorithm progresses, the number of non-dominated fronts (denoted as *K* in the previous section) changes at every generation, and so does the maximum rank the solutions possess. During the later stages of evolution, all the solutions may reside in the first front and thus have the largest neighborhoods. This is why we introduce a coefficient α (≤ 1) into Equation (3). Otherwise, all the solutions would have neighborhoods of size [***L*** **-** ***U***] provided that $\eta = 1$ (later we will see that η is always set to 1). Nevertheless, by limiting the largest neighborhood size with a scaling factor α, the algorithm loses its global exploration ability. But such a drawback is avoided by the new population classification method and the scaling factor α becomes unnecessary (α can always be set to 1). Therefore, compared with the original line-up idea adopted in the naive NEES, the new proposed method has at least three advantages:

- It accelerates the convergence rate of the algorithm to the Pareto-optimal solutions;
- It assures that global search takes place at all generations; and
- It is easy to balance the local search and the global search by simply changing one parameter (*r*).

Let us consider the sample problem given in Equation (4) with the same 6 initial solutions from Table III. The first non-dominated front consists of {s1, s2, s3, s4} and the second front consists of {s4, s5}. They make up the sequence { s1, s3, s2, s4, s5, s6}. Note that the positions of s1, s2, s3 and s4 can be exchanged arbitrarily in the sequence, and so can the positions of s5 and s6. Set *r* to 1. Then, each pseudo front should accommodate the same number of individuals. According to the modified population classification approach we have:

Pseudo front 1={s1, s3};

Pseudo front 2={s2, s4};

Pseudo front 3={s5, s6}.

Set *K* to 5 and α to 1.0. Three different neighborhood sizes must now be calculated:

$$\boldsymbol{D}_1(1)= \frac{1\times 1\times \eta^{1-1}}{5}[\,\boldsymbol{U}-\boldsymbol{L}] = \frac{1}{5}\,[\boldsymbol{U}-\boldsymbol{L}]= \frac{1}{5}\left[\begin{bmatrix}1\\3\end{bmatrix}-\begin{bmatrix}0.1\\0\end{bmatrix}\right]=\begin{bmatrix}0.18\\0.60\end{bmatrix}$$

$$\boldsymbol{D}_2(1)= \frac{2\times 1\times \eta^{1-1}}{5}[\,\boldsymbol{U}-\boldsymbol{L}] = \frac{2}{5}\,[\boldsymbol{U}-\boldsymbol{L}]= \frac{2}{5}\left[\begin{bmatrix}1\\3\end{bmatrix}-\begin{bmatrix}0.1\\0\end{bmatrix}\right]=\begin{bmatrix}0.36\\1.20\end{bmatrix}$$

$$\boldsymbol{D}_3(1)= \frac{3\times 1\times \eta^{1-1}}{5}[\,\boldsymbol{U}-\boldsymbol{L}] = \frac{3}{5}\,[\boldsymbol{U}-\boldsymbol{L}]= \frac{3}{5}\left[\begin{bmatrix}1\\3\end{bmatrix}-\begin{bmatrix}0.1\\0\end{bmatrix}\right]=\begin{bmatrix}0.54\\1.80\end{bmatrix}$$

where $\boldsymbol{D}_1(1)$ is assigned to s1 and s3, $\boldsymbol{D}_2(1)$ is assigned to s2 and s4, and $\boldsymbol{D}_3(1)$ is assigned to s5 and s6. The solutions with their neighborhoods are illustrated in Figure 5. It can be seen that although the neighborhood size of solutions s1 and s3 remains the same as before, the solutions s2, s4, s5 and s6 have greater sized neighborhoods than before.

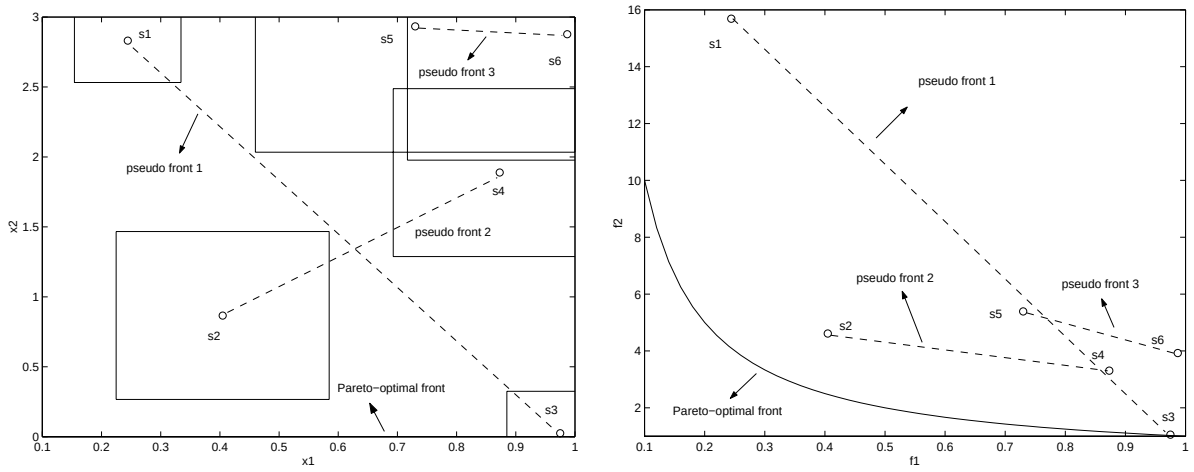

FIGURE 5 Reclassification of the initial population for the sample problem

(Left: decision variable space; Right: objective function space)

5.2 Competition within the individual families

In LCA, the $(1+\lambda)$ individuals compete with each other and there is only one survivor. This competition has two goals. First, it fixes the size of the population at every generation. Second, it prevents premature convergence as it allows low-quality solutions to enter the next generation. But in the case of multi-objective optimization, this competition becomes meaningless, since we use the truncation operator to fix the population size and we can use different levels for the neighborhood exploration mechanism to promote a global search. What we care most is that non-dominated solutions with respect to the whole population are retained. However, we are not interested in distinguishing some solutions within any given family. Thus, there is no need to keep this competition mechanism.

5.3 Neighborhood contraction

As random mutation is inefficient in approaching the precise optimum, LCA utilizes a neighborhood contraction technique, which is adopted in the naive NEES, too. However, we argue that when dealing with multi-objective

optimization problems, it is difficult to define an appropriate contraction function as well as its corresponding parameters unless the contraction procedure has some self-adaptive properties. For instance, if Equation (3) is adopted, the choice of an appropriate value for η is a problem. Consider the sample problem defined in Equation (4). Set K=10, λ=2, η=0.8. Figure 6 shows the output of the NEES with the contraction operator and without the contraction operator after 50 generation with a population size 20. It is observed that with the neighborhood contraction operator, the NEES approach can not converge to the Pareto-optimal solutions when using η=0.8. Another observation is that, if η is set to 0.9, the non-dominated solutions produced by the algorithm can get close to the true Pareto front but they always gather together in a region that covers only a small portion of the Pareto optimal front (results are not shown here for the sake of brevity). So, when dealing with multi-objective optimization problems, this operator affects (in a negative way) either the convergence of the algorithm or the diversity of the solutions obtained.

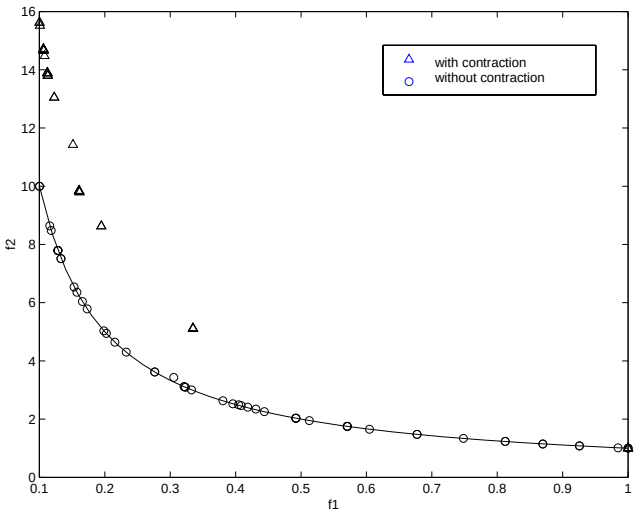

FIGURE 6 The results of the naive NEES with and without the neighborhood contraction on the sample problem

Other contraction methods may be proposed. However, it is very likely that any alternative proposal will require a careful fine-tuning of its parameters. Thus, we decided not to invest any efforts in this direction, and we simple discarded the contraction operator. This decision, however, led us to another important change in the algorithm. We realized that it is unrealistic to expect an evolutionary algorithm to obtain solutions with a precision within 10^{-6} (or higher) in a small number of generations if we only used random mutation on a population of fixed size. To overcome this problem, we decided to introduce a crossover operator. For that sake,

the *simulated binary crossover* (SBX) was incorporated in our proposal [20]. This choice is based on two reasons. First, the two offspring generated by two parents using the SBX operator have a high probability of being close to their parents. The fact that the two offspring produced are likely to lie on the neighborhoods assigned to their parents is in agreement with the principle of neighborhood exploration. Second, there is previous evidence of a very good performance of SBX is operator in diverse numerical optimization tasks [20].

It is however noticed that for most discrete optimization problems, the lack of exploitation capabilities of the mutation operator may not be an important limitation. In fact, we will see later how our NEES can successfully solve a discrete optimization problem without using a crossover operator.

The structure of the modified NEES is outlined in Table VI and all its procedures are graphically illustrated in Figure 7.

TABLE VI The Modified Neighborhood Exploring Evolution Strategy

Initialize the parents population P_1 of size N within the entire search space, and evaluate the population

Perform the non-dominated sorting procedure on P_1

Repeat

- Classify P_t ($t \geq 1$) into a predefined number of subsets (pseudo fronts) and assign each solution a pseudo rank
- Assign neighborhoods to the solutions in P_t according to their pseudo ranks
- Each solution in P_t generates λ offspring within its assigned neighborhood
- All the offspring constitute the offspring population Q_t of size $\overline{N}$, ($\overline{N} = \lambda N$)
- Perform the crossover operation on Q_t
- Combine P_t and Q_t into a mating pool C_t
- Perform the non-dominated sorting procedure on C_t and reduce the size of C_t from $\overline{N}$ to N together with some crowding technique
- Let $P_{t+1} = C_t$, $t = t+1$

Until the terminal condition is met

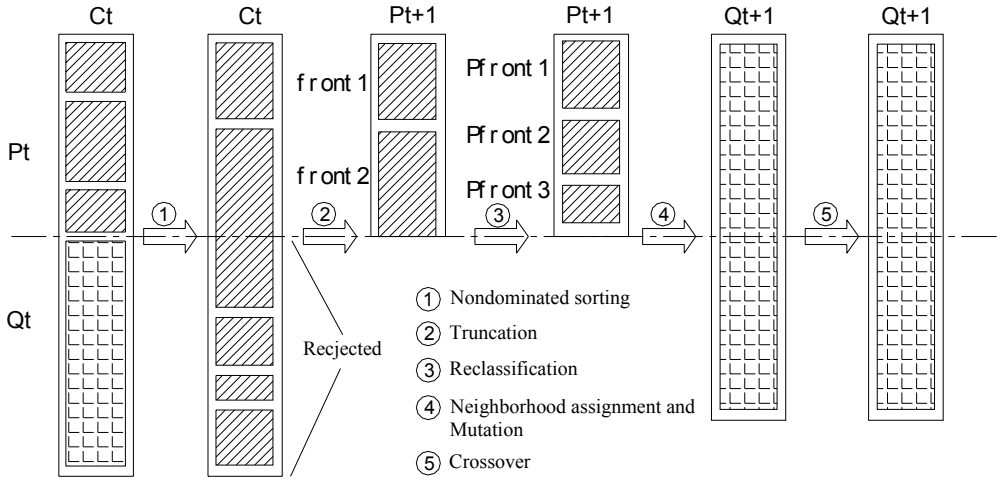

FIGURE 7 Illustration of the modified NEES working process

In the remaining part of this paper, when using the acronym NEES we will be referring to the modified NEES depicted in Table VI instead of its naive version (described Table II) unless otherwise specified.

5.4 Complexity

The basic operations and their worst computational complexities are as follows (suppose the population size is N):

- modified population classification, $O(N)$
- neighborhood assignment, $O(N)$
- nondominated sorting, $O(M\overline{N}^2)$ (see [14])
- crowding distance calculation, $O(M\overline{N}\log\overline{N})$ (see [14])

The overall complexity of NEES is $O(M\overline{N}^2)$, where M is the number of objectives and $\overline{N}$ $(=(1+\lambda)N)$ is the number of solutions in the mating pool, which is governed by the non-dominated sorting part of the algorithm. Note that the overall complexity of NSGA-II is also $O(M\overline{N}^2)$, since it is governed by the non-dominated sorting part, although in the NSGA-II, $\overline{N}=2N$. Hence for $\lambda>1$, the NEES requires a higher computational time at every generation than the NSGA-II.

6 SIMULATION STUDY

In this section, we first test the NEES on a set of optimization problems having two objectives, followed by the simulation study on two problems with more objectives and a discrete optimization problem. For the NEES, we have chosen a reasonable set of values for the control parameters such as K and r and have made little effort in finding the most suitable values for them.

6.1 Two-Objective Optimization Problems

A. Introduction of the problems

Zitzler et al. [18] proposed six test problems, named ZDT1 to ZDT6, which have different difficulty levels, and that have been adopted by a number of other researchers to validate new multi-objective evolutionary algorithms. All the problems are presented in TableVII, except for ZDT5, which is a Boolean function defined over bit-strings since this problem will be discussed later on. In addition, another two problems called *biased test problem* (BTP) and *Griegwank test problem* (GTP) taken from [19] are also tested here. They have the same form of ZDT4 except that different functions $g(\mathbf{x})$ are used. For GTP, the function $g(\mathbf{x})$ has 163^9 local Pareto fronts. For BTP, $g(\mathbf{x})$ is only a monotonic function but it has the property of having more solutions away from the Pareto front, thus difficulting convergence towards the true Pareto front.

For each problem listed in Table VII, both objectives are to be minimized, and the exact global Pareto front corresponds to $0 \le x_1^* \le 1$ and $x_i^* = 0$, i=2, 3, …, n.

TABLE VII Test Problems with Two Objectives

Problem	n	Variable bounds	Objective functions	Properties
ZDT1	30	[0,1]	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1 / g(\mathbf{x})}\right]$ $g(\mathbf{x}) = 1 + (9/(n-1))\sum_{i=2}^{n} x_i$	convex
ZDT2	30	[0,1]	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (x_1 / g(\mathbf{x}))^2\right]$ $g(\mathbf{x}) = 1 + (9/(n-1))\sum_{i=2}^{n} x_i$	non-convex
ZDT3	30	[0,1]	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1 / g(\mathbf{x})} - (x_1 / g(\mathbf{x}))\sin(10\pi x_1)\right]$ $g(\mathbf{x}) = 1 + (9/(n-1))\sum_{i=2}^{n} x_i$	convex and disconnected
ZDT4	10	$x_1 \in [0,1]$ $x_i \in [-5,5]$ i=2,…,30	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1 / g(\mathbf{x})}\right]$ $g(\mathbf{x}) = 1 + 10(n-1) + \sum_{i=2}^{n}\left(x_i^2 - 10\cos(4\pi x_i)\right)$	convex and numerous local traps
ZDT6	30	[0,1]	$f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (f_1(\mathbf{x}) / g(\mathbf{x}))^2\right]$ $g(\mathbf{x}) = 1 + (n-1)\left(\left(\sum_{i=2}^{n} x_i\right) / (n-1)\right)^{0.25}$	non-convex and non-uniformly spaced

GTP	10	$x_1 \in [0,1]$ $x_i \in [-512,512]$ i=2,…,30	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1-\sqrt{x_1 / g(\mathbf{x})}\right]$ $g(\mathbf{x}) = 2 + \sum_{i=2}^{n} x_i^2 / 4000 - \prod_{i=2}^{n} \cos(x_i / \sqrt{i})$	convex and numerous local traps
BTP	30	[0,1]	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1-\sqrt{x_1 / g(\mathbf{x})}\right]$ $g(\mathbf{x}) = 1 + 9\left(\frac{(\sum_{i=2}^{n} x_i - \sum_{i=2}^{n} x_i^{\min})}{(\sum_{i=2}^{n} x_i^{\max} - \sum_{i=2}^{n} x_i^{\min})}\right)^{0.25}$	convex

B. Parameter settings

Currently, the NSGA-II is one of the most competitive multi-objective evolutionary algorithms available and is commonly used as a reference for comparing new algorithms. It has been indicated that the NEES shares much similarity with the NSGA-II except for the mechanism for generating new solutions. Actually, the NEES can be regarded as a variant of the NSGA-II. Thus, a comparison of the two algorithms is mandatory in order to assess the usefulness of the proposed approach.

In this study, for continuous variable optimization problems, the real-coded NSGA-II adopts the *simulated binary crossover* (SBX) and *parameter-based mutation*, which have been found to be superior to other recombination operators [20]. The crossover probability p_c and the mutation probability p_m are set to 0.9 and 1/n, separately, where n is the number of decision variables. The distribution indices for the two operators are set as follows: η_c=20, η_m=20. (All these parameter values are similar to those used in [14]).

For the NEES, there are three required parameters, K, r and λ, which denote the number of pseudo fronts, the reduction ratio in Equation (6) and the number of offspring that each parent generates, respectively. In this study, we set K to 10, and λ to 1, 2 and 4. Consequently, there are three versions of the NEES to be investigated: (1+1)-NEES, (1+2)-NEES and (1+4)-NEES. Later on, we will tune r on a problem to obtain a reasonable value to match the other two (fixed) parameters. The mutation probability p_m (applied at a gene level) is set to 1/n. For the SBX operator in the NEES, the crossover probability p_c and the distribution index η_c are set to 0.9 and 20, respectively. Note the three above parameter values are set exactly equal as in the NSGA-II.

For all the four algorithms, the non-dominated solutions at the end of 200 generations are used for comparison. However, different population sizes should be used here to make a fair comparison. Thus, we set the population

size to 100 for the NSGA-II and (1+1)-NEES, 50 for (1+2)-NEES and 25 for (1+4)-NEES. With such a population setting, all the algorithms will perform the exact same number of fitness function evaluations.

C. Performance measures

As it has been pointed out at the end of Section 2, there are two main goals in multi-objective optimization: the convergence to the Pareto-optimal set and the preservation of the diversity of the solutions. It is difficult to measure these two issues adequately with one performance metric. Here we define three metrics to directly evaluate these two goals.

As the exact Pareto-optimal set of each problem in Table VII is known, a direct method to measure the quality of the solutions obtained is to measure how far the non-dominated set obtained by such algorithm (denoted by Q) is from the exact Pareto-optimal set (denoted by P). The average Euclidean distance of the solutions in Q from the true Pareto-optimal set P can be used for this purpose, which is defined as follows:

$$D=\frac{\sum_{i=1}^{|Q|} d_i}{|Q|} \tag{7}$$

where, d_i stands for the distance of a solution s_i in Q to the Pareto-optimal set P. For each problem in Table VII a large number of uniformly distributed solutions on the known Pareto-optimal front are used to form the Pareto-optimal set P (in this study, 1000 solutions are used). Then, d_i in Equation (7) can be calculated as

$$d_i = \min_{s_j \in P} \left\| s_i - s_j \right\|_2 \tag{8}$$

where s_i and s_j stand for the i-th member of Q and the j-th member of P and $\left\| \bullet \right\|_2$ calculates the 2-norm of a vector. In other words, d_i is the Euclidean distance in the objective space between solution s_i in Q and the nearest member in P.

Additionally to the D metric we used two other metrics, called S and Δ, which are concerned with the diversity of the solutions. Particularly, S measures the extent of spread achieved among the obtained solutions and Δ measures how uniform is the distribution of solutions. Note that the Pareto front of each problem in Table VII (except for ZDT3 which will be specified later on) has two boundary points in objective function space, E_1 (0, 1) and E_2 (1, 0). Thus, in the obtained solution set, we can find two *extreme* solutions s_1^* and s_2^* that are nearest to the two boundary points separately in the objective space. The two particular distances are calculated as follows,

$$d_k = \left\| s_k^* - E_k \right\|_2 = \min_{s_i \in P} \left\| s_i - E_k \right\|_2, \; k=1, 2. \tag{9}$$

Then the *S* metric is defined as

$$S = \sum_{k=1}^{2} d_k \tag{10}$$

Obviously, for a set of the most widely spread non-dominated solutions, *S* would be zero provided that the solutions exactly lie on the true Pareto front. As *S* becomes larger, it means that the non-dominated set has a poorer performance regarding the spread of solutions in objective function space.

The last metric Δ is defined as follows. We sort the obtained non-dominated solutions with respect to the first objective value in an ascending sequence and calculate the Euclidean distance d_i between consecutive solutions in this set. Suppose that there are *N* solutions. Thus, we have (*N*-1) consecutive distances d_i. The metric Δ is defined as the standard deviation of these distances:

$$\Delta = \left(\frac{1}{N-1} \sum_{i=1}^{N-1} (d_i - \overline{d})^2 \right)^{1/2} \tag{11}$$

Thus a lower Δ value indicates a better performance regarding the uniformity of the solutions.

For the three metrics defined above, their values should always be equal to or greater than zero and a smaller indicates a better performance. In evaluating a non-dominated set, these three metrics should be considered simultaneously. Particularly, if one metric gets an extremely poor value, no matter how good the other two metric values are, the non-dominated set is poor.

D. Simulation results

It should be noted that ZDT4 is a rather difficult problem with 21^9 local optima in decision variable space from which only one is obviously the global Pareto front. Arbitrarily, we set the parameters *K*=10 and λ =1 in the NEES for solving this problem. To find a reasonable reduction rate *r* to match *K*, we tuned *r* in the range [0, 1] with an interval of 0.1 and, correspondingly we have 11 NEESs. For each NEES, we performed 30 runs with different initial populations. Figure 8 shows the comparison of the algorithms with respect to the three performance metrics. The box plot is used here for comparing the statistical data. (The box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. The symbol '+' represents the data with values beyond the ends of the whiskers.)

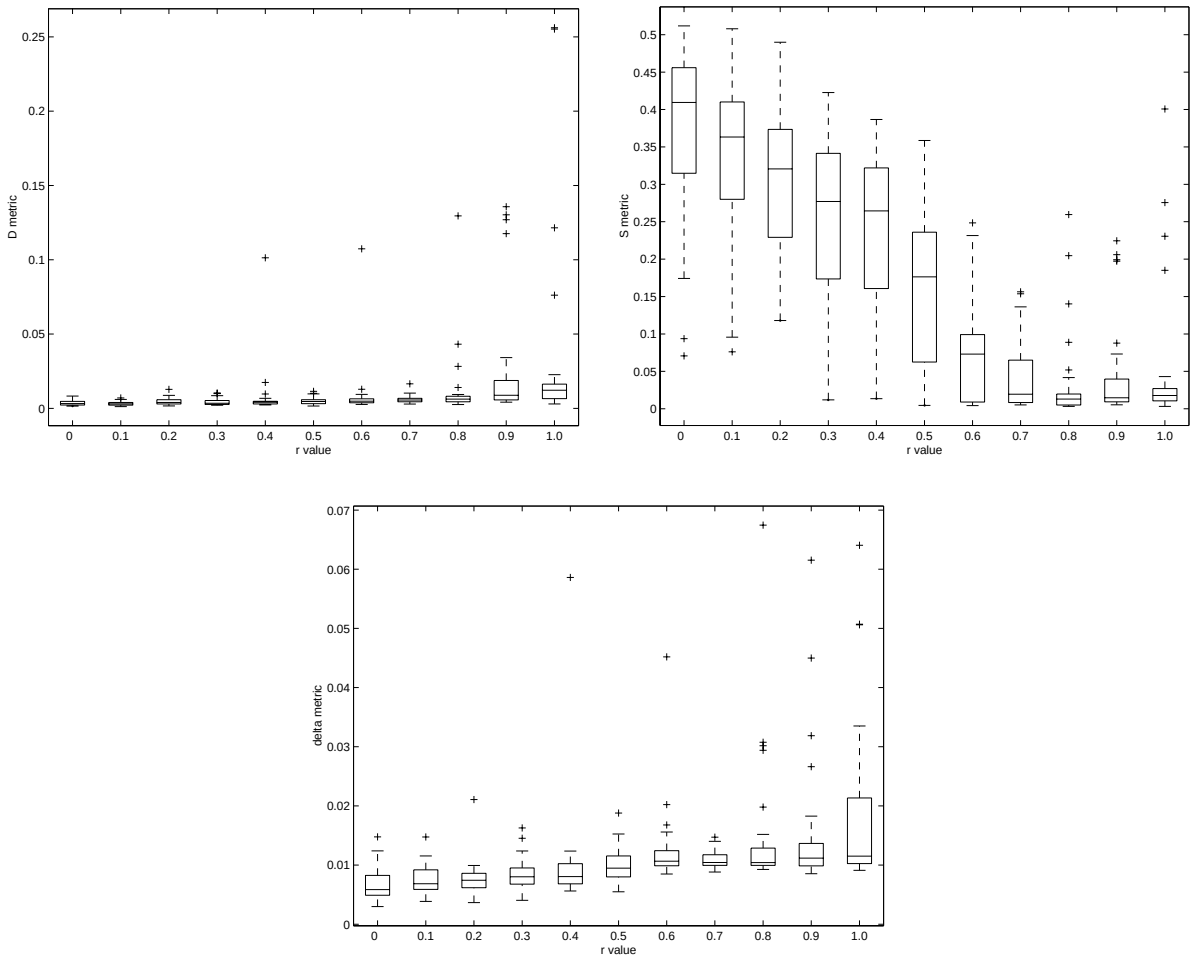

FIGURE 8 The results of (1+1)-NEES on ZDT4 with different reduction ratio *r*

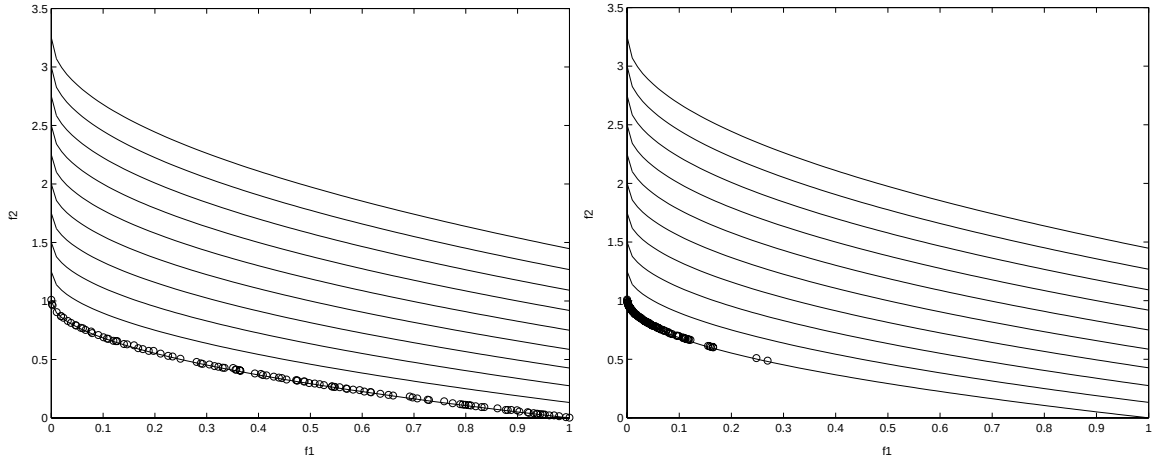

FIGURE 9 Two sets of non-dominated solutions of ZDT6 with different *S* values

It is observed that, for $r \leq 0.8$, the difference among the NEESs in the *D* metric and Δ metric on this problem is not obvious compared with that in the *S* metric. Figure 9 graphically shows the difference between two sets of non-dominated solutions with $S=0.006$ and $S=0.444$, respectively. Based on these observations we find $r=0.8$ is a

relatively good choice for this problem with K=10 and λ=1. We will use K=10 and r=0.8 for all the other problems although it may not be the best choice for every problem.

The mean and variance of the three performance metrics D, S and Δ achieved by the four algorithms on each problem are shown in Table VIII, Table IX and Table X, respectively. It is observed from the tables that the NSGA-II achieved the best convergence on ZDT1 compared with the three NEESs, while (1+1)-NEES obtained the best S values and Δ values (mean and variance). But the overall differences between the two algorithms with respect to the three metrics are small. It can be seen that the (1+1)-NEES has a similar overall performance than the NSGA-II and they both outperform the (1+2)-NEES and the (1+4)-NEES on this problem. Figure 10 shows one of the 30 runs of (1+1)-NEES.

TABLE VIII Mean (First Rows) and Variance (Second Rows) of the *D* Metric

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	GTP	BTP	DTLZ2	DTLZ3
NSGA-II	**0.0006**	**0.0002**	0.0022	0.0066	0.0271	**0.0297**	0.0100	**0.0091**	1.0848
	0.000092	**0.000192**	0.000237	0.0062	0.0030	**0.0101**	0.0012	**0.0013**	1.3047
(1+1)-NEES	0.0009	0.0007	0.0024	0.0123	**0.0040**	0.0334	**0.0004**	0.0136	1.0030
	0.000134	0.000090	0.000148	0.0236	**0.0045**	0.0124	**0.0001**	0.0025	1.4712
(1+2)-NEES	0.0015	0.0011	0.0024	0.0112	**0.0105**	**0.0292**	**0.0005**	0.0170	2.4008
	0.000404	0.000266	0.000219	0.0315	**0.0251**	**0.0113**	**0.0001**	0.0029	2.7779
(1+4)-NEES	0.0030	0.0023	0.0027	**0.0025**	**0.0132**	0.0352	**0.0005**	0.0199	6.4966
	0.000931	0.000735	0.000340	**0.0010**	**0.0347**	0.0139	**0.0001**	0.0054	9.5130

TABLE IX Mean (First Rows) and Variance (Second Rows) of the *S* metric

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	GTP	BTP	DTLZ2	DTLZ3
NSGA-II	0.0069	0.4499	0.3676	0.3932	**0.1688**	0.1730	0.0132	0.0336	0.3939
	0.0059	0.3439	0.0455	0.4194	**0.0033**	0.2711	0.0027	0.0362	0.7858
(1+1)-NEES	**0.0038**	**0.0022**	0.3933	**0.0345**	0.2229	**0.0541**	**0.0013**	0.0459	0.2076
	0.0044	**0.0037**	0.0015	**0.0614**	0.3073	**0.0202**	**0.0038**	0.0479	0.3752
(1+2)-NEES	0.0064	**0.0046**	0.3911	**0.0173**	0.2592	**0.0443**	**0.0004**	0.0335	0.4553
	0.0067	**0.0052**	0.0060	**0.0485**	0.3739	**0.0175**	**0.0005**	0.0224	0.5133
(1+4)-NEES	0.0147	**0.0131**	0.3852	**0.0051**	0.3994	**0.0520**	**0.0014**	0.0673	1.3813
	0.0148	**0.0163**	0.0234	**0.0065**	0.8624	**0.0188**	**0.0029**	0.0736	1.6527

TABLE X Mean (First Rows) and Variance (Second Rows) of the Δ Metric

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	GTP	BTP	DTLZ2	DTLZ3
NSGA-II	0.0100	**0.0036**	0.0287	0.0177	**0.0056**	0.0166	0.0078	0.0378	3.1954
	0.0011	**0.0048**	0.0023	0.0579	**0.0006**	0.0248	0.0007	0.0030	5.5194
(1+1)-NEES	**0.0085**	0.0087	0.0253	0.0149	0.0210	0.0116	0.0095	0.0356	4.6986
	0.0004	0.0005	0.0005	0.0117	0.0541	0.0018	0.0006	0.0031	6.9443
(1+2)-NEES	0.0161	0.0174	0.0345	0.0224	0.0344	0.0215	0.0188	0.0472	7.9480
	0.0016	0.0019	0.0016	0.0064	0.0627	0.0030	0.0015	0.0062	14.1890
(1+4)-NEES	0.0282	0.0318	0.0490	0.0385	0.0799	0.0390	0.0343	0.0605	4.4838
	0.0033	0.0040	0.0052	0.0045	0.1837	0.0054	0.0031	0.0106	8.4806

On ZDT2, though the *D* values of NSGA-II indicate that this algorithm has provided the best approximation of the Pareto front in terms of distance, the *S* values show that the NSGA-II failed to find a good spread of solutions in the entire search space. All the three NEESs have performed better than the NSGA-II with the (1+1)-NEES being the best performer. An arbitrary run of (1+2)-NEES, which is the second-best algorithm on this problem, is shown in Figure 11. In fact, the whole population of the NSGA-II often (in 19 of the 30 runs performed) converged to one point in objective function space: (0, 1). However, in the other 11 runs, the NSGA-II exhibited even better performances than NEESs with respect to all of the three performance metrics. This clearly indicates that the behavior of the NSGA-II was not robust in this problem.

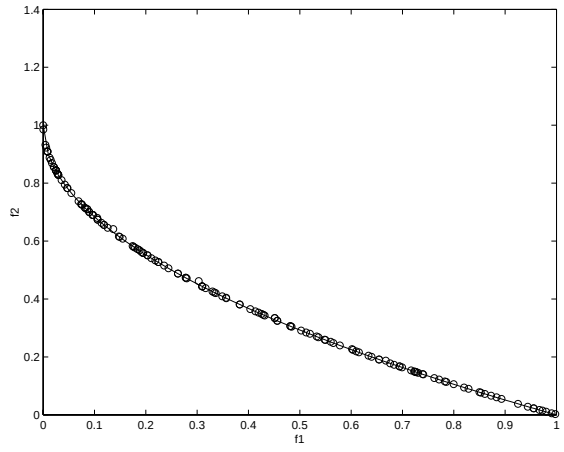

FIGURE 10 Nondominated solutions found with (1+1)-NEES on ZDT1

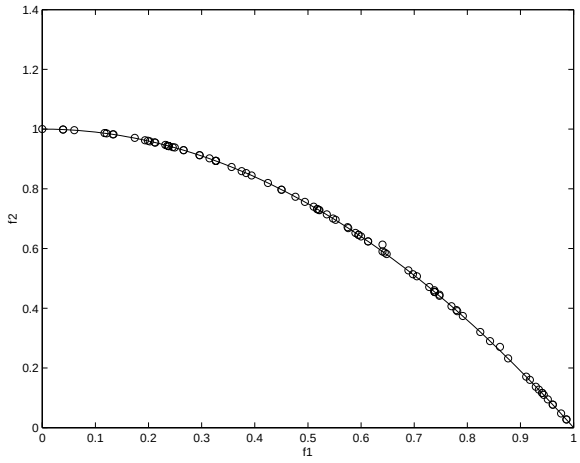

FIGURE 11 Nondominated solutions found with (1+2)-NEES on ZDT2

The Pareto front of ZDT3 is disconnected. Figure 12 shows a typical run of the (1+1)-NEES over 30 runs on this problem. It is obvious that the Δ metric fails to measure the uniformity of the non-dominated solutions correctly. In other words, the Δ values shown for ZDT3 in Table X tell little about the performances of the algorithms. Note that the boundary point E_1 of the true Pareto-optimal set in this problem does lie in (0, 1) while the other boundary point E_2 does not lie in (0, 1) , which is a noticeable difference with respect to the other problems. Nevertheless, the *S* metric defined by Equation (10) still serves as an indirect measure of the maximum space a non-dominated set covers. From Table VIII and Table IX, it is observed that the NSGA-II has performed better than the three NEESs regarding the D and S metrics; however, the differences are very small.

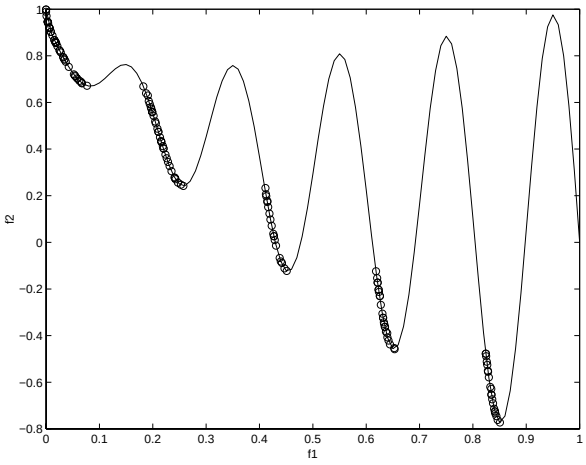

FIGURE 12 Nondominated solutions found with (1+1)-NEES on ZDT3

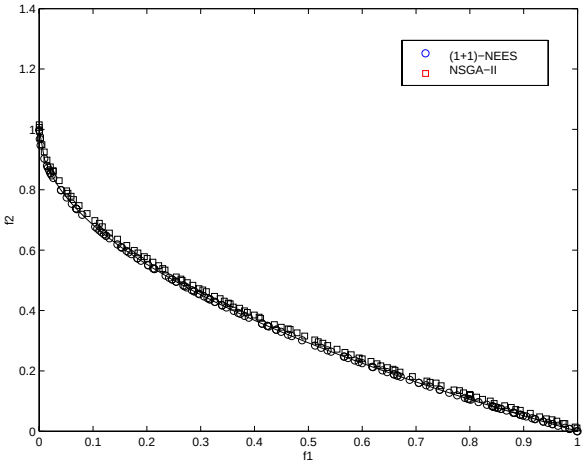

FIGURE 14 Nondominated solutions found with (1+1)-NEES and the NSGA-II on BTP (*D*=0.0004 for NEES and *D*=0.0110 for the NSGA-II)

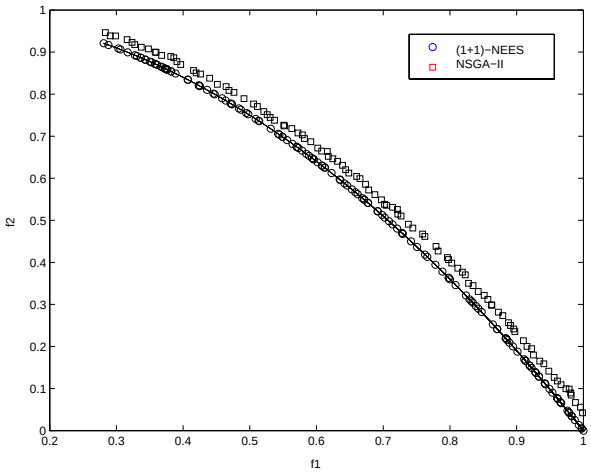

FIGURE 13 Nondominated solutions with (1+1)-NEES and NSGA-II on ZDT6 (D=0.0028 for NEES and D=0.0231 for NSGA-II)

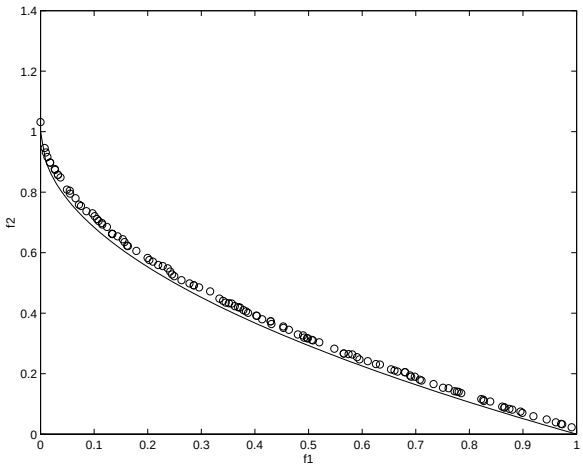

FIGURE 15 Nondominated solutions found with (1+1)-NEES on GTP

It is observed in Table IX that the NSGA-II had a poor performance on ZDT4 regarding the spread metric S (refer to Figure 9 to see different spreads of solution sets with different S values on this problem). Among the NEESs, (1+4)-NEES provided the closest approximation to the true Pareto front. However, when we set the population size of the (1+4)-NEES to 25, the number of non-dominated solutions is far less compared to that obtained in the final population of other algorithms.

It has been observed that on ZDT2 and ZDT4, the neighborhood exploring technique of the NEES exhibits its advantages in extending the range of the non-dominated set. From Table VIII it is also observed that on ZDT6 and BTP the NEES provides a better convergence to the true Pareto front. The mean D values of the three NEESs are less than that of the NSGA-II. Refer to Figure 13 and Figure 14 for typical runs of the (1+1)-NEES and the NSGA-II with different D values on ZDT6 and BTP, respectively. However, on ZDT6 the NSGA-II has the ability to produce more uniformly distributed non-dominated solutions than the NEESs (refer to Table X).

On GTP, with respect to the D metric and Δ metric, the (1+4)-NEES performs obviously worse than others. But the difference among the other algorithms is not significant. Nevertheless, the NEESs always obtain a broader spread of non-dominated sets than the NSGA-II. Figure 15 illustrates the non-dominated solutions obtained by an arbitrary run of the (1+1)-NEES on GTP. We found that over 30 runs on this problem, the NSGA-II obtained several non-dominated sets that covered only about half of the true Pareto front.

6.2 Three-Objective Optimization Problems

A. Introduction of the problems

We consider two problems that have more than two objectives and share the same form [21]:

$$\begin{cases} \min\ f_1(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\cos(x_{M-1}\pi/2) \\ \min\ f_2(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\sin(x_{M-1}\pi/2) \\ \vdots \qquad \vdots \\ \min\ f_M(\mathbf{x}) = (1+g(\mathbf{x}_M))\sin(x_1\pi/2) \end{cases} \tag{12}$$

where, $0 \le x_i \le 1$ (i=1, 2, …, n), $\mathbf{x}_M = [x_M, x_{M+1}, \ldots, x_n]^T$. $g(\mathbf{x}_M)$ must take any function with $g \ge 0$. Let g=0 correspond to the Pareto-optimal front with $\sum_{m=1}^{M} f_m^2(\mathbf{x}) = 1$. Different difficulties can be introduced by defining different $g(\mathbf{x}_M)$ functions. For example,

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 \tag{13}$$

$$g(\mathbf{x}_M) = 100\left(|\mathbf{x}_M| + \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right) \tag{14}$$

where, $|\mathbf{x}_M| = n - M + 1$. The first test problem (called DTLZ2) uses Equation (13) with M=3, n=12, and the second test problem (called DTLZ3) uses Equation (14) with M=4, n=8. Both Pareto fronts correspond to $x_i^* = 0.5$, $i = M, \ldots, n$. As $\sum_{m=1}^{M} (f_m^*)^2 = 1$. For M=3, the desired Pareto front is the first quadrant of a sphere of radius one in R^3 space. And for $M \ge 4$, the desired front is part of a hypersphere of radius one.

B. Parameter settings

All the parameters of the algorithms are set the same as before, except for the maximum number of generations, which is now set to 300 instead of 200.

C. Performance measures

The performance measures are defined the same as for the two-objective optimization problems except for a few differences. The distance between the obtained non-dominated set Q and the exact Pareto-optimal set P is also defined as the D metric using Equation (7). However, as P is part of a sphere (or hypersphere) of radius one, the distance from a solution s_i in Q to P can be calculated as follows,

$$d_i = \left(\sum_{m=1}^{M} (f_m^{(i)})^2\right)^{1/2} - 1.0 \tag{15}$$

where M stands for the number of objectives.

The *S* metric defined by Equation (9) and Equation (10) can be easily extended to measure the maximum spread of a non-dominated set for the class of problems formulated by Equation (12). The subscript of d_k in Equation (9) and Equation (10) should now be equal to 1, 2, …, *M*, since the true Pareto-optimal set has *M* (>2) boundary points in objective function space. The coordinates of these boundary points are in the form (0, 0, …, 1, …, 0, 0), which have *M* elements with only one of them equal to 1 and the others equal to zero.

In order to measure the uniformity of a non-dominated set *Q* we assign a distance value to every solution in the set defined by Equation (16),

$$d_i = \min_{j \neq i} \left\| s_i - s_j \right\|_2 \tag{16}$$

where s_i and s_j stand for the *i*-th and the *j*-th member of *Q* and $\left\| \bullet \right\|_2$ calculates the Euclidian distance between the two solutions in the objective space. The Δ metric is defined as the standard deviation of the d_i values as in Equation (11) except for the fact that in this case there are a total of *N* d_i values instead of (*N*-1) values.

D. Simulation results

The statistical data over 30 independent for each problem are shown in Tables VIII to X. On DTLZ2, the NEESs did converge very close to the Pareto front with satisfactory distribution properties, but the NSGA-II performed even better than them with respect to all of the three performance metrics. A typical run of the (1+1)-NEES is shown in Figure 16. DTLZ3 is a rather difficult problem with 11^5 local non-dominated fronts. None of the NEESs converged to the true Pareto front on this problem, and neither did the NSGA-II.

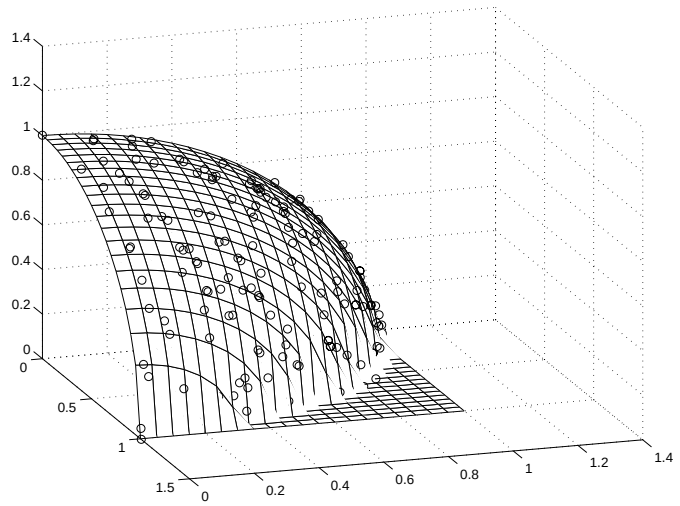

FIGURE 16 Nondominated solutions with (1+1)-NEES on DTLZ2

Based on the comparisons on various continuous optimization problems with two or more objectives, the following issues related to the performances of the algorithms are concluded:

- The NEESs are capable of converging to the true Pareto front with a satisfactory precision considering the great difficulties that several of the test problems adopted have;
- The NSGA-II usually has the ability to converge closer to the Pareto front than the NEESs (though for some problems it does not), while the NEESs have the ability to find a broader spread of non-dominated solutions than the NSGA-II;
- As the value of λ increases, the uniformity of the non-dominated solutions achieved by the NEES decreases (refer to Table X);
- The (1+1)-NEES is better than the (1+2)-NEES and the (1+4)-NEES in terms of the overall performance on most problems.

6.3 Discrete Multi-Objective Optimization Problems

A. Introduction of the problem

It is indicated in Section 6.1 that one of the six functions introduced by Zitzler et al. in [18] is a discrete optimization problem, called ZDT5, which is defined as

$$\begin{cases} \min \quad f_1(\mathbf{x}) = 1 + u(x_1) \\ \min \quad f_2(\mathbf{x}) = g(\mathbf{x}) / f_1(\mathbf{x}) \\ g(\mathbf{x}) = \sum_{i=2}^{n} v(u(x_i)) \\ \forall i = 2, \cdots, n, v(u(x_i)) = \begin{cases} 2 + u(x_i), & if \ \ u(x_i) < 5 \\ 1, & if \ \ u(x_i) = 5 \end{cases} \end{cases} \tag{17}$$

where x_i (i=1,…, n) represents a binary string, $u(x_i)$ gives the number of ones in x_i, n=11, $x_1 \in \{0, 1\}^{30}$ and x_2, , …, $x_n \in \{0, 1\}^5$. The true Pareto corresponds to g(x)=10. Thus, each of the five elements in x_i (i=2,…, n) should be one but there is no requirement on x_1. There are a total of 31 *different* solutions in the Pareto-optimal set considering that the values of $u(x_1)$ should be integers in the range [0, 30]. Here *different* refers to objective function values. Actually, different x_1 may correspond to equal values of $u(x_1)$ and, subsequently, the same point in the objective function space.

B. Special design of the algorithms and parameter settings

Let the 11 x_i variables constitute a vector y. Then, y has a total of 80 (=30+10*5) Boolean numbers and can be regarded as a solution to the problem or a chromosome in the evolutionary population. In order to use the NEES to

solve this problem, the neighborhood as well as the size of a solution should be defined first. Let V^m denote a vector space as follows,

$$V^m = \left\{ \mathbf{y} = (y_1, \cdots, y_m)^T \mid y_i \in \{0,1\}, i = 1, \cdots, m \right\}$$

A transformation ξ_p is defined from $V^m \to V^m$ as: any consecutive $m*p$ elements of $\mathbf{y} \in V^m$ carry out the *xor* operation with a vector consisting of $m*p$ ones, where $p \in (0, 1)$ is a probability. After this operation, all the chosen consecutive $m*p$ elements in $\mathbf{y}$ change their previous values, that is, 0 to 1 and 1 to 0.

Definition 5 (Neighborhood and neighborhood size in V^m) Let $\mathbf{y} \in V^m$. The neighborhood of $\mathbf{y}$ is defined with respect to a probability p as

$$\mathcal{N}(\mathbf{y}) = \left\{ \mathbf{y}' \mid \mathbf{y}' = \xi_p(\mathbf{y}) \right\} \tag{18}$$

The size of the neighborhood is defined as the transformation probability:

$$D = |\mathcal{N}(\mathbf{y})| = p \tag{19}$$

Unlike the neighborhood defined in $\mathbf{R}^n$ that is a hyper rectangle, the neighborhood defined here is a nondeterministic set. And the size of the neighborhood is no longer a vector as in Equation (2), but a scalar.

The linear assignment of the neighborhood sizes along the pseudo Pareto fronts is used again, but unlike in Equation (2), the neighborhood sizes are determined from the first pseudo front to the last pseudo front, which is as follows,

$$D_k = k \cdot D_0 = k \cdot p_0 \quad (k=1, 2, \ldots, K) \tag{20}$$

where k is the pseudo front counter and $p_0=3/m=3/80$. It should be noted from the above equation that the neighborhood sizes do not tune dynamically during the evolutionary progress.

For the NEES, two of the key control parameters K and r remain as 10 and 0.8, respectively, while $\lambda = 1, 2, 4$. For the NSGA-II, the one-point crossover and the bit-wise mutation are used with probabilities 0.9 and $1/m$ respectively, the same as before. The population sizes for the four algorithms also remain the same as before. The non-dominated solutions obtained at the end of the 200th generation are used for comparison.

C. Simulation results

Considering that the Pareto-optimal set of this problem has a finite number of different solutions in objective function space, a direct method to measure the performance of an algorithm is to count the number of the exact Pareto-optimal solutions the algorithm has found after a couple of generations. This is the only criterion used for

our comparison in this study. Ten simulations with different initial populations for each algorithm were analyzed. Table XI indicates the average number of exact Pareto-optimal solutions each algorithm found over ten runs. It is observed that none of Pareto optimal solutions was found by the NSGA-II over ten runs. On the other hand, the NEESs performed quite well on this problem considering that even the (1+4)-NEES whose population size is only 25 has found more than 25 different Pareto-optimal solutions on average. The non-dominated solutions produced by one of the ten runs of (1+1)-NEES are illustrated in Figure 17.

TABLE XI The Average Number of Pareto Solutions Found by Several Algorithms

NSGA-II	(1+1)-NEES	(1+2)-NEES	(1+4)-NEES
0	27.4	26.8	25.5

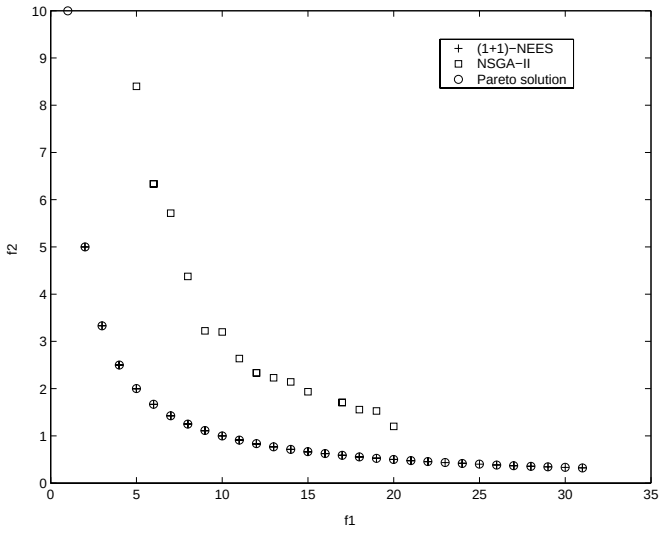

FIGURE 17 The (1+1)-NEES found most of the Pareto-optimal solutions on ZDT5 while the NSGA-II was unable to find any

To validate the efficiency of the neighborhood exploring technique in the NEES, some changes were made to the NSGA-II for further comparison. First, we investigated if it was the bit-wise mutation what led to the poor performance of the NSGA-II by replacing it with the mutation method used in the NEES that is defined as a mathematical transformation. In other words, the NSGA-II now mutates $m*p_m$ genes in each chromosome consecutively (m is the length of the chromosome and p_m is the mutation probability) instead of mutating the randomly chosen genes in the chromosome with a probability p_m. Second, considering that in the NEES the neighborhood sizes for a population are taken from the set $\{3/m, 6/m, 9/m, \ldots, 30/m\}$, we wanted to analyze if the

impact of the mutation operator in the NSGA-II was affected by this fact. Thus, we allowed p_m to vary from 3/m to 30/m with an interval of 3/m, and for each p_m , we executed the NSGA-II ten times. The results turned out to be that none of the 100 runs found any true Pareto-optimal solution. An arbitrary run of the NSGA-II with p_m=3/m is illustrated in Figure 17. Finally, we set the crossover probability p_c of the NSGA-II to zero, making the mechanism for generating new solutions in the NSGA-II the same as that in the NEES. However, the effect of the mutation operator in the algorithms is different. In the NSGA-II, a fixed probability p_m is assigned to each member of a population to produce its offspring, while in the NEES different probabilities from 3/m to 30/m are assigned to the members of a population. Ten runs were performed for the resulting NSGA-II with any of the mutation probabilities in the set {3/m, 6/m, 9/m, …, 30/m}. But again, no Pareto-optimal solutions were found by the NSGA-II in any run. From this study, we conclude that it is the combination of the neighborhood mechanism and the mutation setup of the NEES which contributes to its success in the discrete optimization problem.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced the first proposal to use the *line-up competition algorithm* (LCA) for multi-objective optimization. The proposed algorithm is called *neighborhood exploring evolution strategy* (NEES). This algorithm is based on the non-dominated sorting and diversity preservation ideas, like most of other evolutionary multi-objective optimization algorithms, but it incorporates a key characteristic of the LCA, that is, the different neighborhoods exploration, which distinguishes it from other multi-objective evolutionary algorithms. A comparative study that follows the methodology normally adopted in evolutionary multi-objective optimization [7] was used to validate the viability of our proposal. Three versions of the NEES (a (1+1)-NEES, a (1+2)-NEES and a (1+4)-NEES) were tested and compared to the NSGA-II, which is an algorithm representative of the state-of-the-art in evolutionary multi-objective optimization. The results indicate that the NEES is able to converge to the true Pareto-optimal front with well distributed solutions in objective function space in several problems. The performance of the NEES is competitive with respect to the NSGA-II and, in some cases, it even outperforms it in terms of the spread of solutions achieved. It is however noted that the NSGA-II tends to produce better approximations in terms of closeness of its solutions to the true Pareto fronts of the problems adopted in our study. This sort of behavior is exhibited in most of the problems with continuous decision variables. However, in

the last test problem (which has Boolean decision variables), the NEES has a much better performance (in all its versions) than the NSGA-II. Among the three NEES versions tested, the (1+1)-NEES has the best overall performance.

Several paths of future work are possible. We indicated that one of the major components of the LCA is its mechanism to contract the neighborhoods generation by generation. However, this mechanism could not be successfully adapted in the NEES and further study in this regard is necessary. Some mechanisms of self-adaptation used with evolution strategies may be introduced for this sake. However, the adaptation of such mechanisms to multi-objective optimization problems is not trivial and deserves a serious study.

Finally, the success of the NEES in discrete optimization problems is something that certainly deserves further attention. For example, we intend use the NEES in multi-objective combinatorial optimization problems.

Acknowledgements

The authors thank the valuable comments from the anonymous reviewers which greatly helped them to improve the contents of this paper. The first and third authors acknowledge that this work was partially supported by the National Natural Science Foundation of China (Grant No.60204001). The second author acknowledges support from the mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through project no. 42435-Y.

References

1. Miettinen, K. (1998). *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, Boston, Massachusetts.
2. Nam, D. and Park, C. H. (2000). Multiobjective simulated annealing: A comparative study to evolutionary algorithms. *International Journal of Fuzzy Systems*, 2(2), 87-97.
3. Doerner, K., Gutjahr, W. J., Hartl, R. F., et al. (2001). Ant colony optimization in multiobjective portfolio selection. In: *Proceedings of the 4th Metaheuristics International Conference*. Porto, Portugal, 243-248.
4. Coello, C. A. C. and Cortés, N. C. (2002). An approach to solve multiobjective optimization problems based on an artificial immune system. In: Timmis, J. and Bentley, P. J. (eds), *First International Conference on Artificial Immune Systems (ICARIS'2002)*, University of Kent at Canterbury, England, September, 212-221
5. Hu, X. and Eberhart. R. (2002). Multiobjective optimization using dynamic neighborhood particle swarm

optimization. In: *Proceedings of the 2002 Congress on Evolutionary Computation*, Hawaii, USA, 1677-1681.

6. Coello, C. A. C. and Landa Becerra, R. (2003). Evolutionary multiobjective optimization using a cultural algorithm. In: *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE Service Center, Indianapolis, Indiana, USA, pp. 6-13.
7. Coello, C. A. C., Lamont, G. B. and Van Veldhuizen, D. A. (2002) *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, Boston.
8. Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms*. Ph.D. Thesis, Vanderbilt University, Nashville, USA.
9. Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester: John Wiley & Sons, Ltd.
10. Yan, L. (1998). *New algorithm of global optimization for process system— line-up competition algorithm*. Ph. D. Thesis, Beijing University of Chemical Technology, Beijing. (in Chinese)
11. Yan, L. and Ma, D. (2001). Global optimization of non-convex nonlinear programs using line-up competition algorithm. Computers & Chemical Engineering, 25(11-12), 1601-1610.
12. Yan, L. (2003). Solving combinatorial optimization problems with line-up competition algorithm. *Computers & Chemical Engineering*, 27(2), 251-258.
13. Yan, L. (1999). Solving traveling salesman problem with line-up competition algorithm. *Operational Research and Management Science*, 8(3), 24-30. (in Chinese)
14. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182-197.
15. Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multi-modal function optimization. In: Grefenstette, J. (ed), *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Lawrence Erlbaum Associates, 41-49.
16. Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation*, 3(4), 257-271.
17. Knowles, J. D. and Corne, D. W. (2000). Approximating the non-dominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2), 149-172.
18. Zitzler, E., Deb, K. and Thiele, L. (2000). Comparison of multi-objective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2), 125-148.

19. Deb, K. and Goel, T. (2001). Controlled elitist non-dominated sorting genetic algorithms for better convergence. In: Zitzler, E. et al. (eds), *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001).* Zurich, Switzerland, March: Springer-Verlag, Berlin, 67-81.
20. Deb, K. and Agrawal. R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, 9(2), 115-148.
21. Deb, K., Thiele, L., Laumanns, M., et al. (2002). Scalable multi-objective optimization test problems. In: *Proceedings of Congress on Evolutionary Computation (CEC2002).* Hawaii, USA. IEEE Service Center, 825-830.
22. Jensen, M.T. Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms, *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 5, pp. 503--515, October 2003 .