DNA Computing Based Multi-objective Genetic Algorithm

In this paper, DNA computing based non-dominated sorting genetic algorithm is described for solving the multi-objective optimization problems. First, the inconsistent multi-objective functions are converted into Pareto rank value and density information of solution distribution. Then, the archive is introduced to keep the Pareto front individuals by Pareto sorting, and the maintaining scheme is executed to maintain the evenness of individual distribution in terms of individual crowding measuring. Finally, the gene-level operators of DNA computing are adopted to enhance the global searching capability of a multi-objective genetic algorithm (MOGA). The convergence speed is analyzed, and several suggestions on parameter setting are given based on the convergence analysis. Six multi-objective numeric functions are given, and the application results have shown the efficiency of DNA-MOGA in the evenness of population distribution and the convergence near the Pareto frontier.

4.1 Introduction

Multiple noncommensurable and simultaneously competing objectives are involved in many multi-objective optimization problems. If preference articulation of multiple objectives is aggregated into a scalar function with adequate weights, the multi-objective optimization problem can be transformed into a single-objective optimization problem, which can be solved by many methods [1]. However, various objectives often conflict with one another, and it is often unrealistic to obtain a single optimal solution for a multi-objective optimization problem. Hence, a group of compromise solutions will be derived through a multi-objective optimization algorithm, then, the decision maker will select one among those representative solutions.

Since Sckaffer first proposed Vector Evaluated Genetic Algorithm (VEGA) in 1985 [2], multi-objective evolution algorithms (MOEAs), such as PAES [3], SPEA2 [4], NSGA-II [5], etc., have gained significant attention from various fields. As the Pareto frontier is a set of solutions for multi-objective optimization problems, there

J. Tao et al., *DNA Computing Based Genetic Algorithm*,
https://doi.org/10.1007/978-981-15-5403-2_4

are two targets when applying the multi-objective optimization algorithm, that is, converge to the Pareto frontier and solution diversity preservation. Therefore, the individual fitness value according to the Pareto dominated relationship, as well as individual density information, is calculated by MOEAs. The corresponding individual maintaining and updating strategies are also studied in-depth. Usually, the individual updating strategy is implemented according to Pareto dominated relationship. In the cases that the individuals do not dominate each other, the individual density information considering the diversity index should be used. Though many schemes have considered the evenness degree of individual distribution, there are still short of the concrete measurement index. Some existing indexes are too complex to be applied easily in MOEA. Knowles et al. proposed an adaptive grid archiving strategy in PAES that provably maintained solutions in some "critical" regions of the Pareto front set once they were found [3]. Deb et al. computed the evenness of individual distribution using the cluster analysis method, however, the computing complexity was $O(N^3)$ with population size N [6].

As for theoretical research of MOEA, though most of MOEAs obtained satisfactory results of test functions, they lack theoretical analysis to guarantee the convergence of the solution distribution. Because of the importance of convergence analysis, theoretical research is carried out gradually. Rudolph proved that MOEA converged to the Pareto-optimal set with probability 1 [7]. The convergence of MOEA was usually analyzed by the Markov chain model without consideration of the evenness of individual distribution [8]. Laumanns et al. introduced the concept of ε-dominance and established MOEAs with the desired convergence and distribution properties, however, the convergence of MOEA was not analyzed [9]. Zitzler raised a theoretical analysis method to evaluate the performances of MOEA [10].

In this chapter, a DNA computing based non-dominated sorting multi-objective genetic algorithm (DNA-MOGA) is designed that guarantees both the progress towards the Pareto-optimal set and the evenness distribution of whole non-dominated solutions. The convergence of DNA-MOGA is analyzed based on Markov chain model to prove the effectiveness of the algorithm theoretically.

First, the inconsistent multi-objective fitness functions are converted into a single-objective function by Pareto sorting and individual crowding distance measuring.

Second, an external archive is introduced to keep the Pareto front individuals, and a maintaining scheme is used to maintain the evenness of individual distribution.

Third, the gene-level operators of DNA computing are adopted to enhance the global searching capability of DNA-MOGA.

Finally, six typical multi-objective test functions are applied to show an evenness distribution of Pareto frontier and a quick convergence to the true Pareto-optimal set.

4.2 Multi-objective Optimization Problems

Without loss of generality, multi-objective optimization algorithm seeks to optimize a vector of noncommensurable and competing objectives or cost functions, and the constrained multi-objective optimization problem is described as follows:

$$\begin{aligned} \min \quad & f(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})] \\ s.t. \quad & g_i(\mathbf{x}) \leq 0, \ i = 1, 2, \ldots, h, \ \mathbf{x} \in R^n \end{aligned} \tag{4.1}$$

where f_i $(1 \leq i \leq m)$ is the objective function with totally m objectives, $g_i (1 \leq i \leq h)$ is the constraint condition, and $\mathbf{x} = [x_1, x_2 \ldots x_n]$ is the solution vector to be solved with n variables. Solutions of the multi-objective optimization problem are a family of points known as the Pareto-optimal solution set, where each objective component of any point along with the Pareto frontier can only be improved by degrading at least one of the other objective components [6]. In the absence of preference information of the objectives, the Pareto ranking scheme is regarded as an appropriate approach to represent the strength information of each individual in a MOEA [7]. The vectors are compared according to the dominance relationship defined below [11].

Definition 4.1 (Dominance relationship) a vector $\mathbf{x}$ dominates another vector $\mathbf{y}$, denoted as $\mathbf{x} \prec \mathbf{y}$, if

$$f_i(\mathbf{x}) \leq f_i(\mathbf{y}), \ \forall i \in \{1, 2, \ldots, m\} \tag{4.2}$$

$$f_j(\mathbf{x}) < f_j(\mathbf{y}), \ \exists \ j \in \{1, 2, \ldots, m\} \tag{4.3}$$

Based on the concept of the dominance, the Pareto set can be defined.

Definition 4.2 (*Pareto set*) Let $F \subseteq R^n$ be a set of vectors, then the Pareto set F^* of F is defined as follows: F^* contains all vectors $x \in F$ that are not dominated by any other vector $f \in F$, i.e.

$$F^* := \{x \in F | \nexists \ f \in F : f \prec x\} \tag{4.4}$$

Vectors in F^* are called the Pareto vectors of F. All Pareto set of F is denoted as $P^*(F)$. Moreover, for a given set F, the Pareto set F^* is unique. Therefore, we have $P^*(F) = F^*$. Since the Pareto set F^* is of substantial size for multiple sets of F, the determination of F^* is quite difficult.

To illustrate the concept of Pareto-optimal solution set, the Pareto ranking scheme for a bi-objective minimization problem is shown in Fig. 4.1. As can be seen, it assigns the same smallest ranking value 1 for all non-dominated vectors, while the dominated ones are inversely ranked according to how many individuals in the population dominated them. The Pareto ranking was first introduced by Goldberg [12], and successfully applied by NSGA-II, which ensures that all the non-dominated individuals in the population will be assigned rank 1 and removed from a temporary

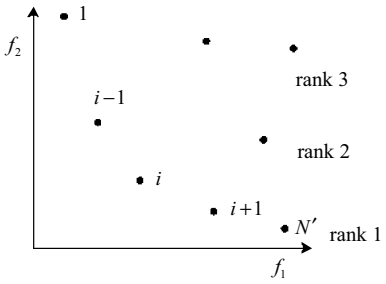

Fig. 4.1 Schematic illustration of Pareto ranking

assertion, then a new set of non-dominated individuals will be assigned 2, and so forth.

4.3 DNA Computing Based MOGA (DNA-MOGA)

Generally, the purpose of MOEA is to find or approximate the Pareto set and keep the diversity of Pareto-optimal solutions [1]. Hence, Pareto ranking and density values of the individual distribution are utilized as two important attributes to each individual. In this chapter, a fitness assignment strategy is designed and an external population is utilized in order to search for an approximated optimal Pareto frontier, and the maintaining scheme is also designed to keep the diversity of Pareto set. In addition, the DNA gene-level operators are introduced to improve the global searching capability of MOGA. Four crucial strategies to improve the performance of DNA-MOGA are discussed as follows.

4.3.1 RNA Encoding

RNA nucleotide base in Chap. 2 is also adopted, 0123 instead of AUGC is used to encode the variable to be solved, and the length of each variable is set as l. Since there are n variables in Eq. 4.1, the length of one chromosome becomes $L = nl$. Moreover, fitness values of multi-objective functions are calculated to analyze the ranking and density values of the chromosome. Thus, the structure of a chromosome is obtained as shown in Fig. 4.2. For a general multi-objective optimization problem, the quaternary encoding chromosome with the length of nl is generated randomly for n variables, Pareto sorting rank and density information are recorded in the $(nl + 1)$

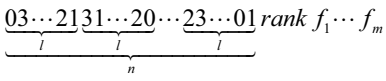

Fig. 4.2 The encoding and structure of one chromosome

th location, and the fitness values of m objective functions are also recorded in the chromosome. The total length of a chromosome is then $nl + m + 1$, where the former nl uses the quaternary encoding and the latter $m + 1$ decimal numbers are obtained in terms of the former quaternary encoding. The decoding process of quaternary encoding is the same as that in Chap. 2.

$$\underbrace{\underbrace{03\cdots21}_{l}\underbrace{31\cdots20}_{l}\cdots\underbrace{23\cdots01}_{l}}_{n} rank\ f_1\cdots f_m$$

4.3.2 *Pareto Sorting and Density Information*

The rank value of Pareto sorting for the individuals has been described in Fig. 4.1. However, the ranking method may fail when most of the individuals do not dominate one another, i.e., all individuals have rank 1. Therefore, additional density information is incorporated to discriminate among individuals with identical rank value. Thus, any multi-objective optimization problem can be converted into a bi-objective optimization problem, i.e., minimizing the ranking value and maximizing the crowding distance [5], which can be further changed into a single-objective optimization problem through some modifications.

In Fig. 4.3, the size of individuals in the Pareto frontier with rank 1 is N' and the ith individual is alongside the Pareto frontier. The calculation of the density information for the ith individual for a bi-objective optimization problem is described as follows:

$$d_i' = 1 \Big/ \sum_{j=1}^{2} (f_j^{i+1} - f_j^{i-1}) \tag{4.5}$$

According to Eq. 4.5, the density value is inversely proportional to the perimeter of a rectangle composed of the dotted line, so the more crowding the individual distribution, the larger the individual density value. For the boundary individuals (1th and N'th), the rectangle is regarded as infinite and the density value is set to 0.

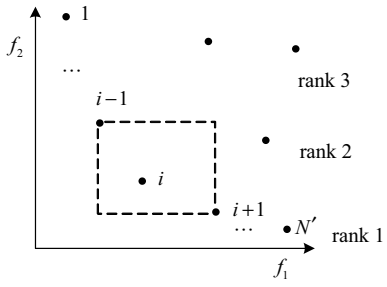

Fig. 4.3 Schematic illustration of individual density calculation

After a simple transformation, e.g., reciprocals operation in Eq. 4.5, the maximization crowding distance is changed into the minimization problem that is consistent with the rank value. As the minimum of Pareto sorting rank is 1, d_i' is normalized to $d_i = d_i' / d_{\max}$, where $d_{\max}$ is the maximum of the current d_i'. Hence, a single-objective fitness function can be derived:

$$F_{fit}(i) = i_{rank} + \lambda d_i \tag{4.6}$$

where i_{rank} is the Pareto sorting rank obtained by non-dominated sorting algorithm that requires $O(mN'^2)$ comparisons [5], and $\lambda = 0.99$ is a coefficient to guarantee $F_{fit}(i)$ less than $i_{rank} + 1$. Equation 4.6 changes the multi-objective optimization problem into a single-objective optimization problem by combining the rank value together with the density information. The calculation complexity of density value is dependent on the sorting algorithm for the current Pareto frontier, and the sorting algorithm requires $O(mN' \log N')$ computations. Hence, the overall computational complexity of the fitness calculation is $O(mN'^2)$.

Matlab Pseudocode of non-dominated sorting algorithm can be obtained from [5], and the code can be gained from their website. All the individuals in the first front are given rank 1, the second front individuals are assigned rank 2, and so on. After assigning the rank, the crowding in each front can be calculated in terms of Eq. 4.6.

4.3.3 Elitist Archiving and Maintaining Scheme

In terms of Eq. 4.6, the individuals satisfying $F_{\text{fit}} < 2$ are obviously the elitists, which will be kept at an archive. At each generation (g), a Pareto approximated set F will be produced and stored to the archive. Thus, the archive size will increase with the evolution process, and it may be far larger than N. Obviously, too many solutions cannot help decision maker deal with the problem. The desirable solution set is an approximation of F^* that dominates all elements of F and of bounded size. Hence, the maximal size of the elitist archive is limited to N. If the size of elitist archive (N') is larger than N, the maintaining scheme for the archive size limitation is implemented to keep the individual diversity and evenness distribution of the individuals. The following principles are abided by: if the new individual dominates partial individuals in the archive, then, the dominated ones are eliminated from the archive and the new individual is added, else, if the individuals do not dominate one another, the maintaining scheme is performed to make the archive size not larger than N and keep the evenness of individual distribution in the archive. Whether it can be added to the archive is depending on the Pareto rank. The computational complexity of the maintaining scheme is similar to the Pareto non-dominated sorting algorithm depending on the size of the elitist archive, i.e., $O(mNN')$. When all individuals in the archive are along the Pareto frontier, a modified adaptive cell density evaluation scheme is implemented that is originated from [13], as shown in Fig. 4.4.

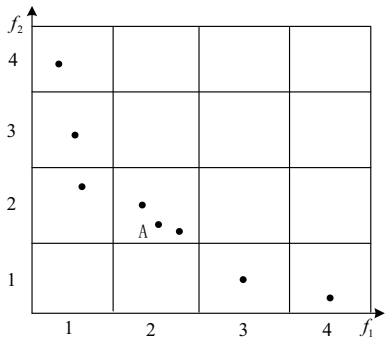

Fig. 4.4 Density map and density grid of the Pareto frontier

After elitist keeping, all individuals in the archive are non-dominated ones. If the archive size is still larger than N, the maintaining scheme is carried out to guarantee that at most one individual is kept at each cell. For example, there are three individuals at cell 'A' in Fig. 4.4, and two individuals will be removed by the maintaining scheme. The cell width in each dimension of objective functions can be calculated as

$$g_{wi} = [\max f_i(\mathbf{x}) - \min f_i(\mathbf{x})] \big/ K_i \tag{4.7}$$

where g_{wi} is the cell width in the ith dimension at generation (g), K_i denotes the number of cells designated for the ith dimension, e.g., in Fig. 4.4, and there exist 16 cells if $K_1 = K_2 = 4$. Since the maximal and minimal fitness values in the objective space will change with the process of evolution, the cell size will vary from generation to generation.

At every generation (g), the cell information is first calculated in terms of Eq. 4.7, and the cells with more than 1 individual will be maintained. The main procedure of the maintaining scheme is given as follows [14].

(1) Calculate the width of the cell (g_{wi}) and add ζ to g_{wi} in order to guarantee that all individuals locate in the cell but are not at the boundary line of the cell.
(2) Repeat to find the position of each individual and judge whether there exist other individuals in the cell; if so, compare the fitness value obtained by Eq. 4.6 and keep the best individual, else the individual is directly added to the archive.
Through the above procedure, there is at most one individual in a cell. The calculation complexity of the maintaining scheme is $O(N'^2)$ to locate the position of the individuals, where N' is the archive size, $N' > N$.
By using the elitist archive and its maintaining scheme, the diversity of the population is kept, and the archive size (N') satisfies $K_j < N' < \sum_{j=1}^{m} K_j$. When there are two or more individuals in a cell, the excessive individuals will be deleted. Matlab code of the maintenance function is given as follows:

```
function CompressedData=maintainf( RawData,ColumnIndex, Count)
% RawData is individuals in the archive, ColumnIndex is nl+M+V, Count is
Population size
[m,n]=size(RawData);
TempPoint=zeros(1,n);
for i=m:-1:2
    for j=1:1:i-1
        if RawData(j,ColumnIndex)>RawData(j+1,ColumnIndex)
            TempPoint=RawData(j,:);
            RawData(j,:)=RawData(j+1,:);
            RawData(j+1,:)=TempPoint;
        end
    end
end
% Locate the cell position
maxX=max(RawData(:,ColumnIndex));
minX=min(RawData(:,ColumnIndex));
maxY=max(RawData(:,ColumnIndex+1));
minY=min(RawData(:,ColumnIndex+1));
 % obtain the length of cell
unitX=(maxX-minX)/Count+1e-4;
unitY=(maxY-minY)/Count+1e-4;
SelectedDataFirst=[zeros(1,ColumnIndex-1),100,100,0,0];
for i=1:m
    areaX=ceil((RawData(i,ColumnIndex)-minX)/unitX+1e-8);
    areaY=ceil((RawData(i,ColumnIndex+1)-minY)/unitY+1e-8);
    [mm,nn]=size(SelectedDataFirst);
    flag=0;
    for j=1:mm
        if SelectedDataFirst(j,nn-1)==areaX &&
SelectedDataFirst(j,nn)==areaY
          flag=1;
            if SelectedDataFirst(j,ColumnIndex)>RawData(i,ColumnIndex)
                &&
                SelectedDataFirst(j,ColumnIndex+1)>RawData(i,ColumnIn
                dex+1)

SelectedDataFirst(j,1:ColumnIndex+1)=RawData(i,:); %COPY the X
information
          end
       end
    end
    if flag == 0

SelectedDataFirst=[SelectedDataFirst;[RawData(i,:),areaX,areaY]];
    end
end
[mm,nn]=size(SelectedDataFirst);
SelectedDataFirst=SelectedDataFirst(2:1:mm,:);
CompressedData=SelectedDataFirst(:,1:ColumnIndex+1);
```

4.3.4 DNA Computing Based Crossover and Mutation Operators

When solving the single-objective optimization problem, various crossover and mutation operators are proposed and the shortcomings of SGA can be alleviated greatly [15]. Recently, DNA computing based gene-level operators have made some significant achievements [16, 17]. However, most of MOEAs utilize the traditional crossover and mutation operators. In this chapter, DNA computing based crossover and mutation operators are utilized to improve the global searching capability of MOGA.

(1) Selection operator

Because the population in the archive is composed of the non-dominated individuals, they are directly selected as the parents of the genetic operators. If the number of the non-dominated individuals N' is less than N, the random selection for a better individual with the value of Eq. 4.6, is adopted to choose the rest $(N - N')$ of the individuals. The Matlab code is given as follows: Note that the proportional selection and Roulette wheel selection can also be used here.

```
function parent_chromosome=chooserest(BestS,SizeBestS,chromosome,pop,BsJi)
% BestS is the individuals in the archive, SizeBestS is N', pop is
population size
if SizeBestS<=pop
      parent_chromosome(1:SizeBestS,:)=BestS;
      kk=SizeBestS+1;
   while kk<=pop
       choose_1=kk;
       choose_2=ceil(rand*pop); % randomly selected individual
       while choose_1==choose_2
          choose_2=ceil(rand*pop);
       end

       if BsJi(choose_1)<=BsJi(choose_2) %better individual is selected.
            parent_chromosome(kk,:)=chromosome(choose_1,:);
            kk=kk+1;
       else
            parent_chromosome(kk,:)=chromosome(choose_2,:);
            kk=kk+1;
       end
    end
  else
     parent_chromosome(1:pop,:)=BestS(1:pop,:);
  end
```

(2) Crossover and mutation operators

Since DNA sequence is made up of four nucleotide bases, i.e., Adenine (A), Uracil (U), Guanine (G), and Cytosine (C). The quadruple encode is also selected as the encoding of chromosomes. Three operations on a single RNA sequence: translocation, transformation, and permutation are adopted as the crossover operators [18]. While three operations for mutation of nucleotide base: reversal, transition, and exchange are utilized as the mutation operators [18]. The corresponding description about RNA crossover and mutation operators, and their Matlab code can refer to Chap. 2.

4.3.5 The Procedure of DNA-MOGA

In DNA-MOGA, a fitness function using Pareto sorting rank with density information is given in Sect. 3.2. The elitist individuals are then kept in the archive. To keep the evenness distribution of the elitist, a maintaining scheme is designed based on the adaptive cell. DNA computing based crossover and mutation operators are introduced to improve the global searching capability of MOGA. The whole procedure of DNA-MOGA is described in the following steps.

Step 1: Initialize the population size N, the number of cells for the ith dimension K_i, and the maximum generation G.

Step 2: Generate the N quaternary encoding chromosomes randomly in the search space, decode, and calculate the fitness value in terms of Eq. 4.6.

Step 3: Keep the elitists in the archive and maintain the archive when the number of archive individuals N' is larger than N.

Step 4: Select the archive population as the parents of the genetic operators. If $N' < N$, use the random selection operation for choosing the rest of the individuals.

Step 5: Carry out the crossover operator in the best $N/2$ individuals in terms of the value of the single-objective function Eq. 4.6, where the permutation operator is implemented with probability 1 and the transformation operator is implemented with probability 0.5. If the transformation operator is not performed, the translocation operator is then implemented. N offspring are generated by the crossover operator.

Step 6: Implement the mutation operator in the left $N/2$ individuals. The nucleotide base is replaced by one of the three integers when the mutation operator is performed.

Step 8: Repeat steps 3–6 until the termination criterion is met, that is, the set maximal generation or the set minimal distance to the true Pareto frontier is satisfied.

4.3.6 Convergence Analysis of DNA-MOGA

At present, the convergence analysis of MOEA mainly lies in the infinite Pareto solutions [9]. In this chapter, the elitists are kept in the archive with bounded size,

and the convergence is analyzed according to the above definition of Pareto non-dominated relationship and the running procedure of DNA-MOGA.

Theorem 4.1 Let $F^{(g)} = \cup_{j=1}^{g} f_i^{(j)}, 1 \leq f_i^{(j)} \leq N', i \in \{1, \ldots, m\}$ be the set of elitists in the archive and maintained by the maintaining scheme. Then $A^{(g)}$ is an approximated Pareto set of $F^{(g)}$ with bounded size $\left|A^{(g)}\right|$, i.e., (1) $A^{(g)} \in P^*(F^{(g)})$, (2) $\left|A^{(g)}\right| \leq \sum_{i=1}^{m} K_i$.

Proof

(1) Suppose $A^{(g)} \notin P^*(F^{(g)})$ is at generation g. According to Definition 4.2, the case occurs only if f is not dominated by any individual of $A^{(g)}$ or not in $A^{(g)}$.

For f that is not in $A^{(g)}$, i.e., f is not an elitist or f is an elitist but removed later on. Removal, however, only takes place when some new f^p enters the archive which dominates f. The case contradicts the assumption that f is not dominated by any individuals of $A^{(g)}$. Likewise, removal takes place in the maintaining scheme when f^p and f are located in the same cell, i.e., both of them are individuals in the Pareto frontier, and the fitness value of f^p is superior to f's, which contradicts with the assumption that f is not in $A^{(g)}$.

For f in $A^{(g)}$, however, f does not belong to the Pareto-optimal set. Hence, there exists $f^p \in F^{*(g)}$ that $f^p \prec f$. If this is the case, f will be eliminated by elitist maintaining scheme, i.e., $f \notin A^{(g)}$, which contradicts the assumption.

(2) In terms of the maintaining scheme, the objective spaces are divided into $\prod_{i=1}^{m} K_i$ cells. For $A^{(g)} \in P^*(F^{(g)})$, only those with rank 1 is maintained, and at most one individual can be kept at each cell. Therefore, the maximum number of individuals distributed at the cells is $\sum_{i=1}^{m} K_i$, i.e., $\left|A^{(g)}\right| \leq \sum_{i=1}^{m} K_i$.

Theorem 4.2 If $\left|A^{(g)}\right| \leq N$, DNA-MOGA converges to the Pareto-optimal set with probability 1.

Proof In terms of theorem 4.1, the size of Pareto-optimal set $\left|A^{(g)}\right|$ is the boundary. If the transition probability matrix (P) is the finite state homogeneous Markov chain, then the state of the population is ergodic. Because the elitist maintaining scheme is adopted in DNA-MOGA, the Pareto-optimal individual can be kept and the dominated ones are eliminated finally.

Since the population size is finite and the crossover and mutation operators with elitist are adopted in DNA-MOGA, the transition probability matrix has been proved as the finite states homogeneous Markov chain in [19], i.e., the whole state of population can be reached by DNA-MOGA. Obviously, $\left|A^{(g)}\right| \leq N$ is the premise of the ergodicity.

In terms of theorem 4.1 and 4.2, DNA-MOGA can not only converge to the Pareto-optimal set with probability 1 but also keep the evenness of the population distribution using maintaining scheme. Though $\left|A^{(g)}\right| \leq N$ is required by the convergence analysis, $\left|A^{(g)}\right| = \sum_{i=1}^{k} K_i$ is an extreme number obtained by DNA-MOGA. Hence, it is not necessary for N to be larger than $\sum_{i=1}^{m} K_i$, however, N should be at least greater than $\max(K_i)$.

4.4 Simulations on Test Functions by DNA-MOGA

4.4.1 Test Functions and Performance Metrics

Six test problems are used to examine the performances of DNA-MOGA as listed in the Appendix, where ZDT3, ZDT4, and ZDT6 were designed by Zitzler, as can be found in [4] and [20], and other test functions include DEB, FON, and KUR problems. These problems have characteristics that are suitable for examining the effectiveness of multi-objective optimization approaches in terms of maintaining the population diversity, as well as converging to the Pareto frontier. Many researchers, such as Zitzler et al. [4], Deb et al. [5], Knowles and Corne [13], and Tan et al. [21], have used these test problems in their research on MOEAs.

Three different quantitative performance metrics for multi-objective optimization approaches are utilized, which are capable of evaluating non-dominated individuals and widely used in the different MOEAs [4, 5, 21].

(1) Generational Distance (*GD*): The metric of generational distance represents how "far" the known Pareto front (PF_{known}) is from the true Pareto front (PF_{true}), which is defined as

$$GD = \left(\frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} d_i^2 \right)^{1/2} \tag{4.8}$$

where n_{PF} is the number of individuals in PF_{known}, and d_i is the Euclidean distance in the objective domain between the individuals in PF_{known} and its nearest individual in PF_{true}. The smaller the value of *GD*, the better the approximation of the Pareto-optimal set becomes.

(2) Evenness Spacing (*ES*): The metric of spacing measures how "even" the individuals in PF_{known} are distributed. It is defined as

$$ES = \frac{\left[\frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} (d_i' - \bar{d}')^2 \right]^{1/2}}{\bar{d}'}, \quad \bar{d}' = \frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} d_i' \tag{4.9}$$

where d_i' is the Euclidean distance in the objective domain between the ith individual in PF_{known} and its nearest one. The smaller the value of *ES*, the more evenness the distribution of Pareto frontier will be.

(3) Maximum Spread (*MS*): The metric of maximum spread measures how 'well' the PF_{true} is covered by the PF_{known} through hyper-boxes formed by the extreme function values observed in the PF_{true} and PF_{known}. In order to normalize the metric, it is described as

$$MS = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left\{\frac{[\min(f_i^{\max}, F_i^{\max}) - \max(f_i^{\min}, F_i^{\min})]}{F_i^{\max} - F_i^{\min}}\right\}^2} \tag{4.10}$$

where m is the number of objectives, $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum values of the ith objective function in PF_{known}; $F_i^{\max}$ and $F_i^{\min}$ are the maximum and minimum values of the ith objective in PF_{true}, respectively. The maximum of MS is 1, which shows that the extremum of each objective function is covered by the solutions in the Pareto frontier.

4.4.2 Calculation Results

When DNA-MOGA is used, quaternary encoding is adopted to simulate four nucleotide bases, the individual length for each variable is set to 10, initial population size N is set to 60, the maximum generation number is limited to 10000, and the cell number for each objective dimension K_i is set to 50. The runs of DNA-MOGA terminate when $SD < 10^{-3}$ is satisfied or the maximum generation is performed. NSGA-II is also used to optimize the same multi-objective optimization problems and compared with DNA-MOGA. NSGA-III has obtained a great improvement in solving the multi-objective problem with more than two objectives [22], here, it is not adopted to solve the bi-objective optimization functions.

Two MOEAs are run 50 times, respectively, and the best results with Pareto frontier are selected as the final results. The simulation results are illustrated in Figs. 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, and listed in Table 4.1. It can be seen that the DNA-MOGA guarantees both progress towards the Pareto-optimal frontier and covering the whole range of non-dominated solutions evenly. For DEB problem and ZDT4

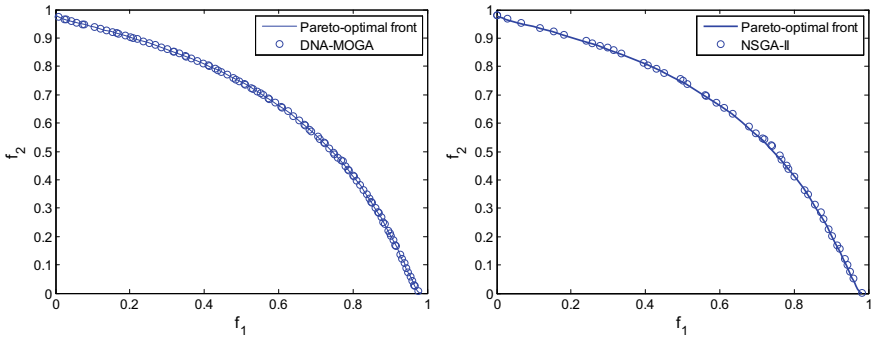

Fig. 4.5 Pareto-optimal solutions to FON problem found by DNA-MOGA and NSGA-II

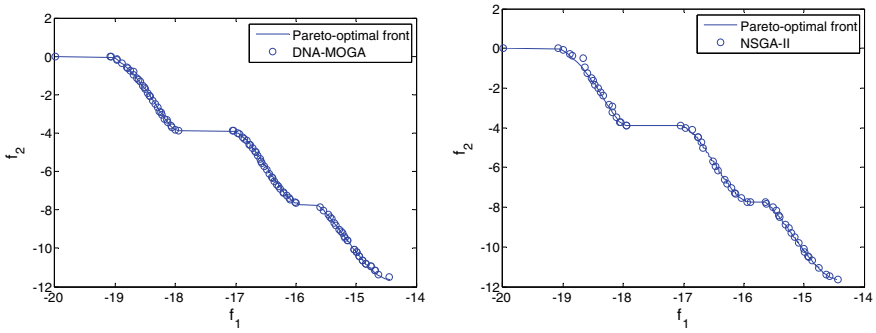

Fig. 4.6 Pareto-optimal solutions to KUR problem found by DNA-MOGA and NSGA-II

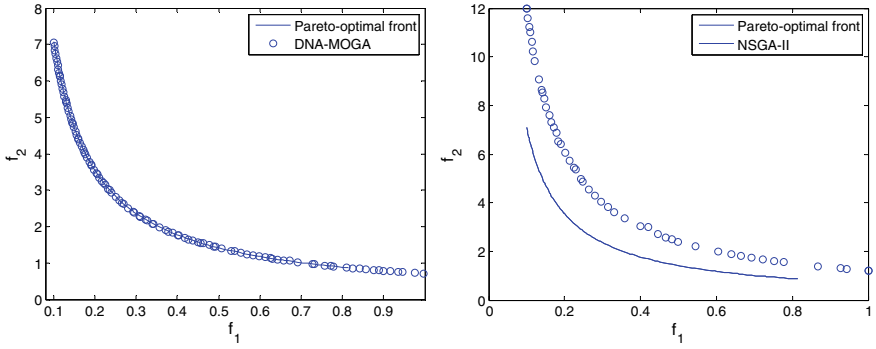

Fig. 4.7 Pareto-optimal solutions to DEB problem found by DNA-MOGA and NSGA-II

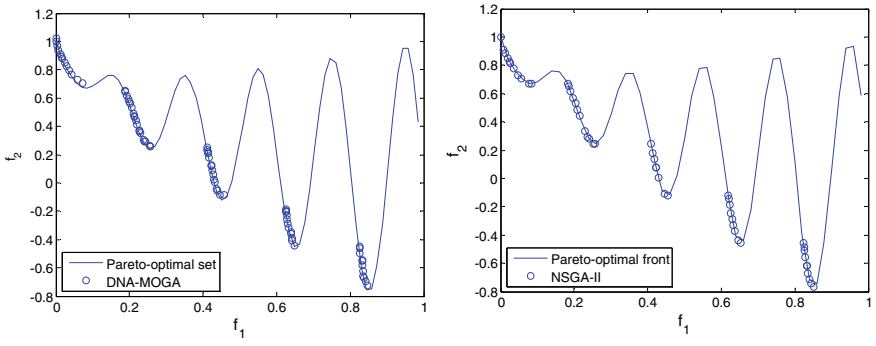

Fig. 4.8 Pareto-optimal solutions to ZDT3 problem found by DNA-MOGA and NSGA-II

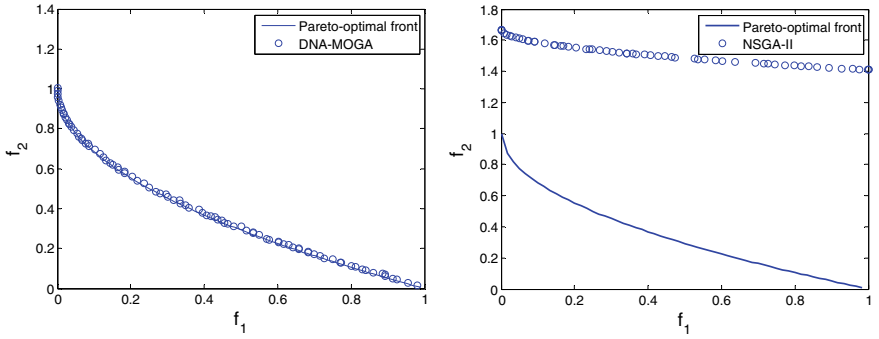

Fig. 4.9 Pareto-optimal solutions to ZDT4 problem found by DNA-MOGA and NSGA-II

Table 4.1 The comparison results for test problems over 50 runs

Test problems	NSGA-II			DNA-MOGA		
	SD	ES	MS	SD	ES	MS
FON	0.0014	0.6236	1.0000	0.0012	0.3032	0.9948
KUR	0.0056	0.6717	1.0000	0.0045	0.9995	0.9935
DEB	0.0723	0.7114	0.7181	0.0056	0.4403	0.9981
ZDT3	0.0069	0.6374	0.9333	0.0058	0.4620	0.9311
ZDT4	0.0956	0.7425	0.7279	0.0021	0.4589	0.9999
ZDT6	0.0069	0.5811	1.0000	0.0049	0.2304	1.0000

problem with many local minima and fraudulence characteristics, the advantages of DNA-MOGA are especially obvious from Figs. 4.7 and 4.9, and the measure of *SD* in Table 4.1 also shows the improvement of DNA-MOGA in approximating to the Pareto-optimal solution set. The metrics of *ES* indicate the superiority of DNA-MOGA in the evenness of population distribution except for the KUR problem.

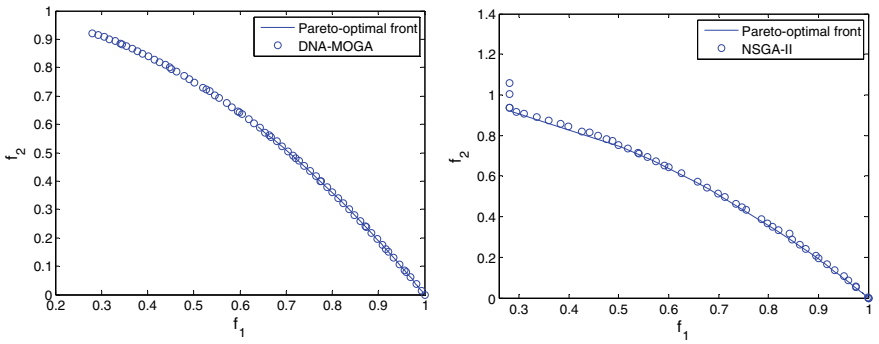

Fig. 4.10 Pareto-optimal solutions to ZDT6 problem found by DNA-MOGA and NSGA-II

Concerning the measure of *MS*, because one cell is required to keep at most one individual, the individual with larger fitness value but better single-objective value will be removed, which may affect the performance of *MS*. The comparison results from Table 4.1 illuminate this deficiency of DNA-MOGA, especially the KUR problem. Since the Pareto front of the KUR problem is noncontinuous and the distance between 2 noncontinuous frontiers is relatively large, the drawback of DNA-MOGA is then prominent and the metrics of *ES* is inferior to NSGA-II's, which can be improved by increasing the number of the cells. However, the computation burden will become heavy with the increasement of the cell number. Since *SD* and *ES* are the main indexes of the multi-objective optimization problem, DNA-MOGA is superior to NSGA-II as a whole.

4.5 Summary

The DNA computing based non-dominated sorting genetic algorithm is suggested for multi-objective optimization problems. The inconsistent multi-objective fitness functions are converted into a single-objective function by Pareto sorting and individual crowding measuring. And an external archive is introduced to keep the individuals in the Pareto frontier, and the maintaining scheme is designed to keep the evenness of individual distribution. The gene-level operators of DNA computing are also adopted to enhance the global searching capability of MOGA. Convergence analysis and simulation results on typical multi-objective test problems show the improvement of DNA-MOGA in the spread of solutions and the convergence near the true Pareto-optimal set.

Appendix

Typical multi-objective test problems

Problems	The minimum objective functions	Optimal solutions	Comments
DEB	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g \Big/ f_1(\mathbf{x})$ $g = 2 - \exp\left\{-\left(\frac{x_2 - 0.2}{0.004}\right)^2\right\}$ $- 0.8\exp\left\{-\left(\frac{x_2 - 0.6}{0.4}\right)^2\right\}$	$x_i \in [0.1, 1]$ $i = 1, 2$	Non-convex
FON	$f_1(\mathbf{x}) = 1 - \exp(-\sum_{i=1}^{3}(x_i - 1\Big/\sqrt{3})^2)$ $f_2(\mathbf{x}) = 1 - \exp(-\sum_{i=1}^{3}(x_i + 1\Big/\sqrt{3})^2)$	$x_i \in [-4, 4]$ $i = 1, 2, 3$	Non-convex
KUR	$f_1(\mathbf{x}) = \sum_{i=1}^{2}[1 - 10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})]$ $f_2(\mathbf{x}) = \sum_{i=1}^{3}[\lvert x_i\rvert^{0.8} + 5\sin(x_i^3)]$	$x_i \in [-5, 5]$ $i = 1, 2, 3$	Non-convex Noncontinuous
ZDT3	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g[1 - (x_1 \Big/ g)^{1/2} - x_1\sin(10\pi x_1)\Big/ g]$ $g = 1 + 9\left(\sum_{i=2}^{n} x_i \Big/ (n-1)\right)$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g[1 - (x_1 \Big/ g)^{1/2} - x_1\sin(10\pi x_1)\Big/ g]$ $g = 1 + 9\left(\sum_{i=2}^{n} x_i \Big/ (n-1)\right)$	$x_i \in [0, 1]$ $i = 1, 2, \ldots, 30$	Convex, disconnected
ZDT4	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g[1 - (x_1 \Big/ g)^{1/2}]$ $g = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10\cos(4\pi x_i)]$	$x_1 \in [0, 1]$ $x_i \in [-5, 5]$ $i = 2, \cdots, 10$	Non-convex
ZDT6	$f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(\mathbf{x}) = g[1 - (x_1 \Big/ g)^2]$ $g = 1 + 9\left(\sum_{i=2}^{n} x_i \Big/ (n-1)\right)^{0.25}$	$x_i \in [0, 1]$ $i = 1, 2, \ldots, 10$	Non-convex non-uniformly spaced

Matlab code of the main function for improved DNA-MOGA to solve the multi-objective function

```
clc;
clear all;
close all;

global howlong G
fig=0;
G=10000;                                % Maximal generation
pop=60;                                 % Population size
V=2;  fmax=0;   M=2;    howlong=8;   num=V;
CodeL=V*howlong;
Rep=1;

amax=ones(1,V);
amin=0.1*ones(1,V);
max_range=amax;min_range=amin;
 for rep=1:1:Rep
    t0=clock;  SizeBestS1=0;deltkg=1;

  chromosome = initialize_variablesRNADEB(pop,M, V, min_range,
max_range);
   chromosome= non_domination_sort_modRNA(chromosome, M, V);
   BsJi=chromosome(:,CodeL+1);
   a=find(BsJi<2);%%%find the non-dominated individuals
   ma=size(a,1);
   BestS=[];
   for i=1:ma
       BestS=[BestS;chromosome(a(i),:)];
   end
kkk=1;
for kg=1:1:G
    time(kg)=kg;
   BsJi=chromosome(:,CodeL+1);
   a=find(BsJi<2);
   ma=size(a,1);
   for i=1:ma
      BestS=[BestS;chromosome(a(i),:)];
   end
   SizeBestS=size(BestS,1);
   tempBest= non_domination_sort_modRNA1(BestS, M, V);
   tempBsJi=tempBest(:,CodeL+1);
       a=find(tempBsJi<2);
       ma=size(a,1); BestS=[];
        for i=1:ma
             BestS=[BestS;tempBest(a(i),:)];
        end
```

```
        SizeBestS=size(BestS,1);
        if SizeBestS>pop
             BestS=SelectComData(BestS(:,1:M+CodeL+1),CodeL+M,60);
            SizeBestS=size(BestS,1);
            deltsize(deltkg)=SizeBestS-SizeBestS1;
             deltkg=deltkg+1;
            SizeBestS1=SizeBestS;
        end
          if deltkg>1201
            sumdeltsize=sum(deltsize(deltkg-1200:deltkg-1));
            if sumdeltsize<1
               break;
            end

       end

    parent_chromosome=chooserest(BestS,SizeBestS,chromosome,pop,BsJi);
     [temp_chromosome1,indexp]=sort(parent_chromosome(:,CodeL+1));
     for i=1:pop
        parent_chromosome1(i,:)=parent_chromosome(indexp(i),:);
     end
     offspring_chromosome = ...
        genetic_operatorRNADEB(parent_chromosome1,i,pop,...
        M, V, min_range, max_range);
     chromosome=non_domination_sort_modRNA(offspring_chromosome, M, V);
end
wholetime=etime(clock,t0)
bestjj(rep,:,:)=BestS;
rep
end
    plot(BestS(:,CodeL + 2),BestS(:,CodeL + 3),'o');
    xlabel('f_1','fontsize',12);
    ylabel('f_2','fontsize',12);
```

References

1. Deb, K. 2001. Multi-objective optimization using evolutionary algorithms. John Wiley & Sons.
2. Schaffer, J.D. 1985. Multiple Optimization with vector evaluated genetic algorithms. In *International conference on genetic algorithms.*
3. Knowles, J.D., and D.W. Corne. 2000. Approximating the nondominated front using the pareto archived evolution strategy, vol. 8.
4. Zitzler, E., K. Deb, and Thiele, L. 2000. Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation* 8 (2): 173–195.
5. Deb, K., and D. Kalyanmoy. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197.

6. Konak, A., Coit D.W., and Smith, A.E. 2006. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety* 91 (9): 992–1007.
7. Rudolph, G. 1998. On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In *IEEE World Congress on IEEE International Conference on Evolutionary Computation.*
8. Rudolph, G.N., and A. Agapie. 2000. Convergence properties of some multi-objective evolutionary algorithms. In *Congress on Evolutionary Computation.*
9. Laumanns, M., et al. 2002. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation* 10 (3): 263–282.
10. Zitzler, E., et al. 2003. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation* 7 (2): 117–132.
11. Fonseca, C.M., and P.J.J.E.C. Fleming. 2014. An overview of evolutionary algorithms in multiobjective optimization. 3 (1): 1–16.
12. Goldberg, D.E. 1989. Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Boston, MA.
13. Knowles, J., and D. Corne. 1999. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of Congress on Evolutionary Computation.*
14. Tao, J., Q. Fan., and Chen X, et al. 2012. Constraint multi-objective automated synthesis for CMOS operational amplifier.*Neurocomputing* 98: 108–113.
15. Michalewicz Z. 2013. Genetic algorithms + data structures = evolution programs[M]. Springer Science & Business Media.
16. Ren, L., et al. 2010. Emergence of self-learning fuzzy systems by a new virus DNA–based evolutionary algorithm. *International Journal of Intelligent Systems* 18 (3): 339–354.
17. Jan, H.Y., C.L. Lin, and Hwang, T.S. 2006. Self-organized PID control design using DNA computing approach. *Journal of the Chinese Institute of Engineers 29* (2): 251–261.
18. Tao, J., and N. Wang. 2007. DNA computing based RNA genetic algorithm with applications in parameter estimation of chemical engineering processes. *Computers Chemical Engineering*31 (12): 1602–1618.
19. Liepins, G.E. 1992. Global convergence of genetic algorithms. Proceedings of SPIE - The International Society for Optical Engineering 1766: 61–65.
20. Fonseca, C.M. and P.J. Fleming 2002. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. II. Application example. *IEEE Transactions on Systems, Man Cybernetics, Part A 28* (1): 38–47.
21. Tan, K.C., Y.J. Yang, and T.H. Lee. 2006. A distributed cooperative coevolutionary algorithm for multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 10 (5): 527–549.
22. Mkaouer, W., M, Kessentini., and Shaout A, et al. 2015. Many-objective software remodularization using NSGA-III. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 24 (3): 1–45.