ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

An Improved NSGA-II Algorithm Based on Novel Rank and Crowded Comparison Selection Operator

Qi Zhao[1, a], Hanning Chen[1]

[1]School of Computer Science and Technology, Tianjin Polytechnic University, Tianjin 300000, China

[a]781081917@qq.com

Abstract

The multi-objective optimization problem (MOP) is currently an important area of research and development. For any multi-objective evolutionary algorithm, increasing the convergence and diversity of the Pareto solution set is two of the most important goals. This paper proposed a novel rank and crowded comparison selection operator to optimize the diversity of the Pareto solution set. First, in this paper we changed the way of crowded distance calculation in the last selected Pareto front. Second, a redundant individual deletion strategy is used to protect the diversity of the solution set. At the same time, in order to improve the convergence of proposed algorithm, this paper introduces a non-uniform mutation operator to replace the polynomial mutation operator used in the original NSGA-II algorithm. The performance of the proposed algorithm was tested by ZDT series test function and compared with other three popular algorithms. Simulation results verify the superiority and effectiveness of the proposed algorithm.

Keywords

Multi-objective optimization; NSGA-II; Non-uniform mutation operator; Congestion comparison selection operator.

1. INTRODUCTION

Most real-world optimization problems involve simultaneously minimizing or maximizing multiple possible conflicting goals while following some constraints. Unlike single-objective optimization, multi-objective optimization problem (MOP) produces a set of solutions that balance each goals, every solutions showing a specific compromise between different goals. Multi-objective evolutionary algorithms are one of the most commonly used strategies for solving complex optimization problems. In the past two decades, domestic and foreign scholars have done a lot of research on multi-objective optimization problems. There are many popular MOEAs that have been proposed, such as NSGA-II[1], SPEA2[2], MOEA/D[3], OMOPSO[4], and SMPSO[5],etc.

From all the mentioned MOEAs, NSGA-II is the most popular MOEA. NSGA-II is designed on the basis of NSGA [6], using non-dominated sorting and elite strategies. The populations neighborhood density is measured according to the crowding distance between individuals. The populations non-dominated rank is defined according to the fitness value of the individual in the population. In order to improve the performance of the algorithm, Deb et al use a selection operator based on non-dominated rank and crowding distance. Non-dominated sorting and crowding distance assignment strategy have been widely used in other MOEAs, such as MODE, MOPSO, etc. Like other MOEA, the improvement goal of NSGA-II is to improve the convergence

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

and diversity of the Pareto solution set. The improved method of NSGA-II based on niche technology is proposed in the literature [7].The improved method of NSGA-II based on minimum spanning tree was put forward in Literature [8].The improved algorithm proposed in Literature [9] has taken Pareto sorting hierarchy and congestion degree of individuals into account. Sindhya et al. [10] use the crowding distance to preserve diversity in a hybrid MOEA framework. It has been proved that non-uniform mutation operators have good convergence to evolutionary algorithms [11]. There are also some studies that improve the cross-operation method of NSGA-II, such as using the principle of adjacent maximal to select the crossover individuals, that is, selecting the individuals with the largest distance between adjacent individuals for hybridization to improve the distribution of Pareto frontier populations [12].It is extremely difficult to produce a system that can withstand harsh environments and has strong adaptability to future generations. Based on this, multi-population genetic algorithms have been proposed and widely developed [13, 14]. Each subgroup performs independent genetic operations to effectively improve population diversity. In order to further improve the population diversity and search performance of NSGA-II, we proposed a novel selection operator to improve the diversity of the algorithm. Meanwhile, the non-uniform mutation operator [15] is introduced into the original NSGA-II algorithm.

The rest of this article is organized as follows: Section 2 highlights the related work. Section 3 proposes an improved NSGA-II algorithm based on a novel rank and crowding distance selection operator (NRCNSGA-II). To demonstrate the excellence of NRCNSGA-II, Section 4 details a series of experimental results and analysis comparisons. Section 5 gives conclusions.

2. RELATED WORKS

2.1. Multi-Objective Optimization

A multi-objective problem optimization problem can be defifined as follows [16]:

$$\begin{aligned} &\min y = F(x) = [\, f_1(x),\, f_2(x), \cdots f_m(x)] \\ &s.t. \quad g_i(x) \le 0, \quad i = 1, 2, \cdots, p \\ &\qquad h_j(x) = 0, \quad j = 1, 2, \cdots, q \end{aligned} \tag{1}$$

In Eq.(1), the F(x) is called the objective function. The gi(x)(i=1,2,...,p) and hj(x)(j=1,2,...,q) is constraint function. where x = (x1, x2, . . ., xn)D is a n-dimensional decision vector and D is a decision space. y = (f1, f2, . . ., fm) Y is a m-dimensional objective vector and Y is an objective space. Fi (x)(i = 1, 2, . . ., m) is the ith objective function to be minimized. In the following, some important definitions of multi-objective problem is given.

Definition 1(Pareto dominance): If decision variable x0=(x1,x2,...xn)D dominates x1=(x1,x2,...xn)D, denoted by x0x1. If and only if:

$$\begin{aligned} &f_i(x^0) \le f_i(x^1), \quad i = (1, 2, \ldots, m) \\ &f_i(x^0) < f_i(x^1), \quad \exists i \in \{1, 2, \ldots, m\} \end{aligned} \tag{2}$$

Definition 2(Pareto optimal solution): If there is no decision vector in the entire parameter space to dominate other decision vector, then the decision vector is called the Pareto optimal solution. All Pareto optimal solutions constitute the Pareto optimal solution set.

Definition 3(Pareto front): The set of all Pareto optimal solutions in the target space is called the Pareto front. Denoted by:

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

$$PF = \{F(x) = (f_1(x), f_2(x), \ldots, f_m(x)) \mid x \in Ps\} \quad (3)$$

Where Ps is Pareto optimal solution.

2.2. Non-dominated Sorting Genetic Algorithm (NSGA-II)

NSGA-II is proposed by K.Deb et al. It is one of the most popular multi-objective optimization evolutionary algorithms. The core of the NSGA-II evolutionary algorithm has two aspects: fast non-dominated sorting and elite selection strategies for all individuals in the population. A fast non-dominated ranking is established based on the two indicators of non-dominated level and crowding distance. The non-dominated level is determined by the Pareto dominance relationship between the optimal performance functions. The main loop of the algorithm are as follows: First, initialize a parent population P(t) of size N, select, cross, and mutate the initial population to form the progeny population Q(t), and combine the two populations to form a scale 2N population R(t). Secondly, the combined population R(t) is sorted and stratified by fast non-domination sort. At the same time, the crowding distance is calculated for each non-dominated layer individual, and the appropriate individuals are selected according to the non-dominated rank and crowding distance to form a new parent pupulation P(t+1). Then generate a new progeny population Q(t+1) by the basic operation of the genetic algorithm, combine P(t+1) and Q(t+1), repeat the above operation until Meet the termination condition of the program. Algorithm 1 gives the main loop of NSGA-II.

	Algorithm 1: NSGA-II main loop
1	While condition:
2	R(t) = P(t) + Q(t)
3	F = fast_nondominate_sort(R(t))
4	P(t+1) = [ ]
5	i = 0
6	While len(P(t+1)) + len(F[i]) < N:
7	Crowding_distance_assignment(F[i])
8	P(t+1) += F[i]
9	i += 1
10	P(t+1) += F[i][0:N-len(P(t+1))]
11	Q(t+1) = make_new_generation(P(t+1))
12	t = t+1

2.2.1. Density Estimation

In order to estimate the density of solutions around a particular point in the population, Deb et al. proposed a new approach. Take the average distance between two points on both sides of the point along each target as the crowded distance of the point. As shown in Figure 1, the croding distance of solution i can be calculated as follows:

$$D(i) = \sum_{m=1}^{M} \frac{F_m^{i+1} - F_m^{i-1}}{F_m^{\max} - F_m^{\min}} \quad (4)$$

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

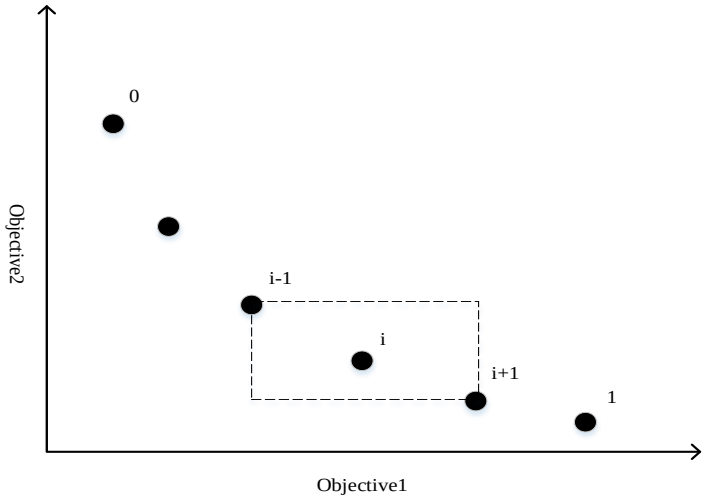

Figure 1. The crowding distance calculation is shown.

When comparing individuals in the same non-dominated rank, we give the ordering sequence of the solutions in that hierarchy based on the solutions fitness value on the mth objective. In Eq.(4), Fi+1 mand Fi-1 m indicates the fitness value of the i+1th and i-1th solution in the sequence of the mth objective. Fmax m and Fmin m is the maximum and minimum values on the mth objective. However, we found that this method of density estimation is not appropriate. Because individuals in the same density estimation area will have the same crowding distance whether it is close to or far from its neighbor solution.

2.2.1. Density Estimation

The crowded comparison operator directs the selection process at various stage of the algorithm to a uniformly distributed Pareto optimal front. A solution with a lower rank is preferentially selected between two solutions having different non-dominated rank. If the two solutions have the same rank, a solution with a larger crowding distance will be selected.

After assigning a non-dominated rank to the solution set by fast non-dominated sorting, different Pareto fronts are generated according to the rank. The original NSGA-II algorithm selects solutions with the same Pareto front by comparing the crowded distances between solutions. Solutions with larger crowding distances are preferentially selected. However, the method based on the crowded distance does not select a different set of solutions from the final selected Pareto front. For example, suppose there are six non-dominated solutions on the last Pareto front to be selected, as shown in Figure 2. It is necessary to select five solutions from the Fareto front to enter the new population. We use the standard NSGA-II method based on the crowding distance, and the results will choose A, C, D, E, F. However, in order to maintain a different set of solutions at the Pareto front, Solution B should be chosen and choose choose D or E to form a new population rather than both D and E. It can be clearly seen from the Figure 2 that the method based on the standard crowded distance will choose two closer solutions. Therefore, this method does not protect diversity in selected solutions. At the same time, we find that when the population size is large, there will be similar solutions or the same solution in the solution set of the same Pareto front, which greatly reduces the diversity of the algorithm.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

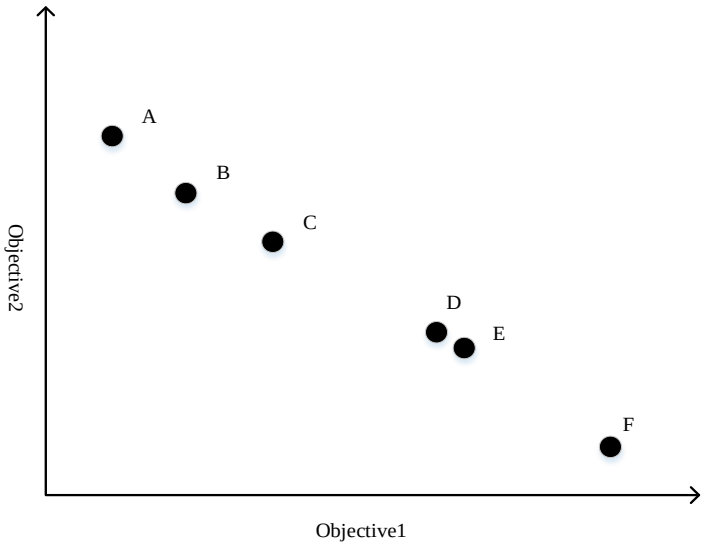

Figure 2. A set of 6 non-dominated solutions from which 5 are selected based on the crowding distance based method used in NSGA-II.

3. PROPOSED WORK

3.1. Non-uniform Mutation Operator

Non-uniform mutations have a refinement ability that depends on the number of populations to achieve a balance between exploration and exploitation. This search is performed uniformly at the beginning and is executed very locally at the end of the search. Michalewicz[15] proposed a dynamic non-uniform mutation operator to reduce the shortcomings of random variation in evolutionary algorithms. For each solution xi tin the t generation population, the offspring xi t+1 are generated by non-uniform variation. If xi t={x1, x2,...xm} is a solution to the t generation and this mutation selected the kth component xk, thus the result of the mutation is xt+1 i={x'1,x'2,...,x'm}, Where

$$x_k^{'} = \begin{cases} x_k + \Delta(t, UB - x_k), & if\ \ a\ random\ \xi = 0 \\ x_k - \Delta(t, x_k - LB), & if\ \ a\ random\ \xi = 1 \end{cases} \tag{5}$$

In Eq.(5), LB and UB are the upper and lower bounds of the variable xk. The function Δ(t,y) returns a value in the range [0,y] such that Δ(t,y) is close to zero as t increases. This property causes this operator to search the space evenly at the beginning of the iteration (when t is small) and to search very locally at a later time. This strategy increases the probability of generating a new number close to its successor than a random choice. We use the following function:

$$\Delta(t, y) = y \cdot (1 - r^{(1-\frac{t}{T})^b}) \tag{6}$$

Where r is a uniform random number from [0, 1], T is the maximal generation number, and b is a system parameter determining the degree of dependency on the iteration number.

3.2. NRCNSGA-II

The original NSGA-II algorithm base on the Pareto front and crowding distance when selecting a non-dominated solution. Between two solutions with different non-dominated rank,

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

we choose a solution with a lower non-dominated rank. If both solution belong to the same Pareto front, then we will prefer to choose the one with a larger crowding distance. Through previous analysis we have found that this choice will undoubtedly reduce the diversity of the solution set. Algorithm 2 is the ranking and crowded comparision selection operator.The improved details are mainly three point:

	Algorithm 2: Rank and Crowded Comparison Selection Operator
1	NewPop = Null
2	rank_index = 0
3	While len(NewPop) < N:
4	If len(Rank[rank_index])<(N-len(NewPop)):
5	If len(NewPop) < (N/4):
6	NewPop.push(Rank[rank_index])
7	rank_index += 1
8	Else:
9	Calculate the crowded distance of solutions in next pareto front use original method
10	Delete solution with the minimum crowding distance
11	NewPop.push(Rank[rank_index])
12	rank_index += 1
13	Else:
14	Calculate the crowded distance of solutions in the pareto front use proposed method
15	For i in range(0,len(Rank[rank_index])-(N-len(NewPop))):
16	Delete solution with the minimum crowding distance
17	Recalculate the crowding distance of the neighbor of deleted solution
18	NewPop.push(Rank[rank_index])
19	Return NewPop

3.2.1. Dynamic Density Estimation

The original NSGA-II algorithm selects the non-dominated solution based on the solution's Pareto rank and the crowding distance. Between two solutions with different non-dominated ranks, we choose a solution with a lower non-dominated rank. If both solution belong to the same Pareto front, then we prefer to choose the one with a larger crowding distance. Through previous analysis we have found that this choice will undoubtedly reduce the diversity of the solution set. Our proposal is as follows:

$$D(i)=\sum_{m=1}^{M}\frac{F_m^{i+1}-F_m^{i}}{F_m^{\max}-F_m^{\min}} \tag{7}$$

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

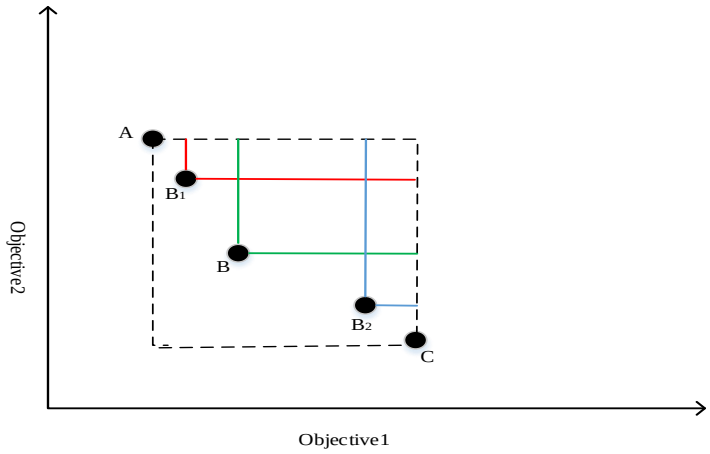

Figure 3. The crowding distance of point B with different positions

As shown in the figure 3, if we use the calculation method of the density estimaton in the original NSGA-II algorithm, the solution B at any position in the cuboid formed by points A and C will have the same crowding distance, whether solution B moves to positon B1 or position B2. obviously this calculation method is not suitable. Because when point B is particularly close to A or C, this will significantly reduce the diversity of the solution set. If we take the sum of the distances from point B to point C in the direction of each objective function as the crowding distance of point B, we can achieve a dynamic estimation as the Fig 3 shows. The area surrounded by the red solid line is the crowding distance of B1. The area surrounded by the green solid line is the crowding distance of B. The area surrounded by the blue solid line is the crowding distance of B2. The crowding distance is changing when point B moves in the cuboid area from A to C. We hope that this way we can achieve a dynamic density estimation of the crowding distance.

3.2.2. Redundant Solutions Elimination Strategy

When we use the fast non-dominated sorting algorithm to assignment rank the Pareto front and then we may found that solutions located in the same Pareto front may be particularly close or even repetitive. This may reduce the diversity of the solution set. In order to overcome this problem, we suggest that until the number of individuals in the new population accounts for a quarter of the maximum number, the solution with the minimum crowding distance in the Pareto front of each rank are deleted. This method can well protect the diversity of the solution set and does not remove the elite solution that plays a key role in the convergence of the algorithm.

3.2.3. Last Front Selection

When selecting a solution based on the crowding distance, we use the method of deleting the solution with minimum crowding distance one by one and recalculate the crowded distance of the deleted solution's neighbor after each delete operation. Calculated as follows:

$$D(i+1)=\sum_{m=1}^{M}\frac{F_m^{i+2}-F_m^{i+1}}{F_m^{\max}-F_m^{\min}} \tag{8}$$

$$D(i-1)=\sum_{m=1}^{M}\frac{F_m^{i+1}-F_m^{i-1}}{F_m^{\max}-F_m^{\min}} \tag{9}$$

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

Where i is the label of each deleted solution.

4. TESTS

4.1. Planning

We selected five well-known ZDT series test functions [19]. Including ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 to test the performance of the proposed algorithm. We set the number of variables n=30, 30, 30, 10, 10 respectively. ZDT 1, ZDT 2 and ZDT 3 have convex, non-convex and discontinuous Pareto fronts. The ZDT 4 problem has 219 different local Pareto fronts in the search space, but only one problem corresponds to the real Pareto front. The ZDT 6 problem has a non-convex and non-uniform Pareto front. The specific form of the test function is shown in Table 1.

Table 1. The test functions of ZDT series

Name	Decision space	Objective function	True Pareto front
ZDT1	$x \in [0,1]^n$ $n = 30$	$f_1(x) = x_1$ $f_2(x) = g(x) * h(x)$ $g(x) = 1 + \frac{9}{(n-1)} * \sum_{i=2}^{n} x_i$ $h(x) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}$	$x_i = 0, i = 2,3,...,30$
ZDT2	$x \in [0,1]^n$ $n = 30$	$f_1(x) = x_1$ $f_2(x) = g(x) * h(x)$ $h(x) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2$	$x_i = 0, i = 2,3,...,30$
ZDT3	$x \in [0,1]^n$ $n = 30$	$f_1(x) = x_1$ $f_2(x) = g(x) * h(x)$ $h(x) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \frac{f_1(x) * \sin(10\pi x_1)}{g(x)}$	$x_i = 0, i = 2,3,...,30$
ZDT4	$x_1 \in [0,1]^n$ $x_i \in [-5,5]$ $i = 2,3,...,n$ $n = 10$	$f_1(x) = x_1$ $f_2(x) = g(x) * h(x)$ $g(x) = 91 + \sum_{i=2}^{n} x_i^2 - 10\cos(4\pi \sum_{i=2}^{n} x_i)$ $h(x) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}$	$x_i = 0, i = 2,3,...,30$
ZDT6	$x \in [0,1]^n$ $n = 10$	$f_1(x) = 1 - \exp(-4x_1)(\sin(6\pi x_1))^6$ $f_2(x) = g(x) * h(x)$ $g(x) = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{(n-1)}\right)^{0.25}$ $h(x) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2$	$x_i = 0, i = 2,3,...,30$

We compared NRCNSGA-II to the standard NSGA-II, MOEA/D and OMOPSO. In the following experiment, the population size was set to 100, the number of iterations was 25,000 and the external archive size was chosen to be 100. The crossover probability of NSGA-II was 0.8, and the mutation rate was the reciprocal of the chromosome length. For each test function, 30

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

independent runs were performed. All runs are performed in the same environment, running on an Intel Corei 5-3230M 2.6 GHz CPU with 4 GB of memory.

There are two goals in MOP: (1) Convergence with the Pareto optimal solution set. (2) Distribution and diversity of non-dominated solutions.

We use the GD indicator [17] to evaluate the convergence of the proposed improved algorithm. Veldhuizen and Lamont proposed the concept of intergenerational distance (GD) and calculated the distance between the non-dominated solution set P and the Pareto optimal front P* measure obtained by the algorithm. It can be defined as:

$$GD = \frac{\sqrt{\sum_{x \in p} \min dis(x, P^*)^2}}{|P|} \tag{10}$$

Where mindis(x, P*) represents the minimum Euclidean distance between the point x in the solution set and the solution in the reference solution set P*. The GD indicator measures the distance between these non-dominated solutions and the solution of the Pareto optimal set. A smaller GD value indicates that it converges better to the Pareto-optimal front.

The Inverted Generational Distance (IGD)[18] is a comprehensive performance evaluation indicator. It evaluates the convergence performance and distribution performance of the algorithm by calculating the minimum distance between each solutions on the real Pareto front surface and the solution set obtained by the algorithm. The smaller the IGD value, the better the overall performance of the algorithm, including convergence and distribution performance. It can be defined as:

$$IGD(P^*, P) = \frac{\sum_{x \in P^*} \min dis(x, P)}{|P^*|} \tag{11}$$

Where P* is the set of solution uniformly distributed on the real Pareto surface, and |P*| is the number of solution of the solution set distributed on the real Pareto surface. P is the optimal Pareto optimal solution set obtained by the algorithm. And mindis(x,P) is the minimum Euclidean distance from individual x to population P in P*. Therefore, the IGD evaluates the overall performance of the algorithm by calculating the average of the minimum distances from the set of points on the real Pareto surface to the acquired population. Through Eq.(11), it can be seen that when the convergence performance of the algorithm is better, the mindis(x, P) is relatively small, so that the convergence performance of the algorithm can be evaluated. however, when the distribution performance of the algorithm is poor, most individuals in the population are Focusing on a small area, we can see that many individuals have large mindis(x, P), so we can evaluate the distribution performance of the algorithm. Smaller IGD values have better convergence and diversity for the Pareto optimal front.

4.2. Results and Analysis

The proposed algorithm is tested on standard test problems ZDT. The results presented in Table 3, Table 4 are the average of 30 runs. We have compared different MOEAs using two metric: GD and IGD. Figures 3-7 show the Pareto front generated by the four algorithms mentioned above.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

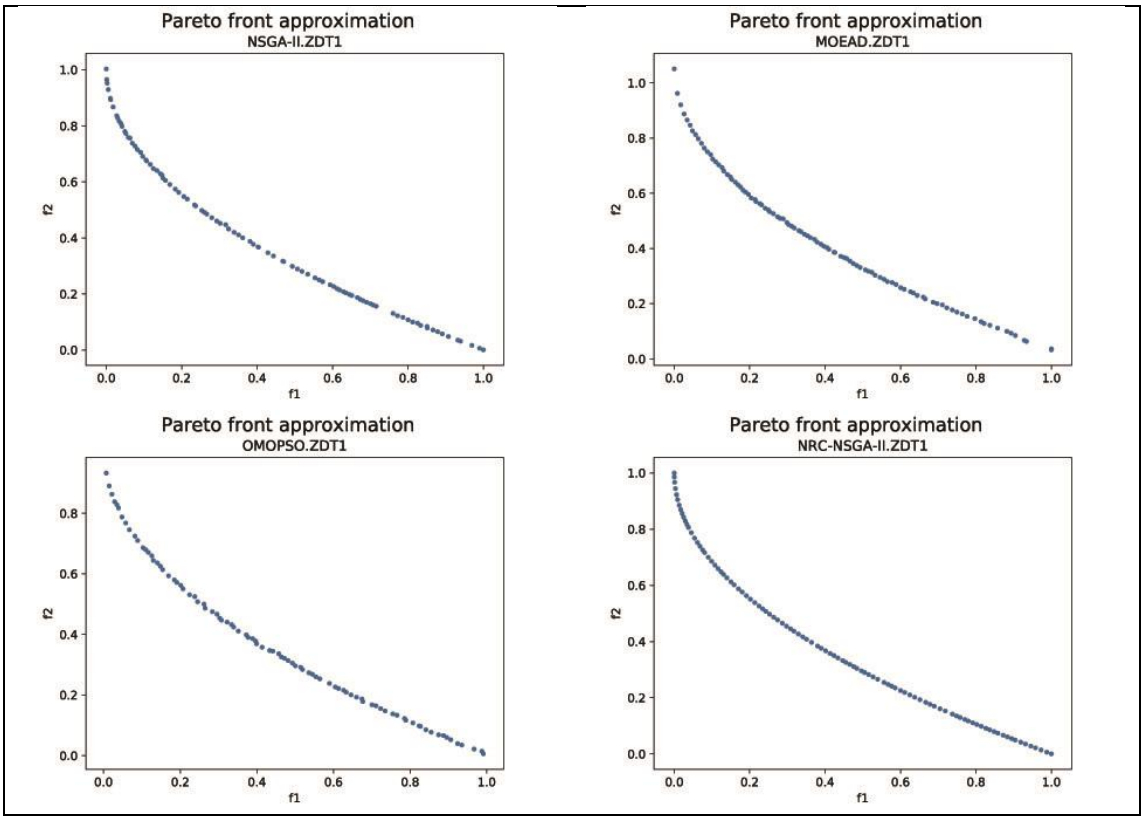

Figure 3. The Pareto fronts produced by four algorithms for ZDT1

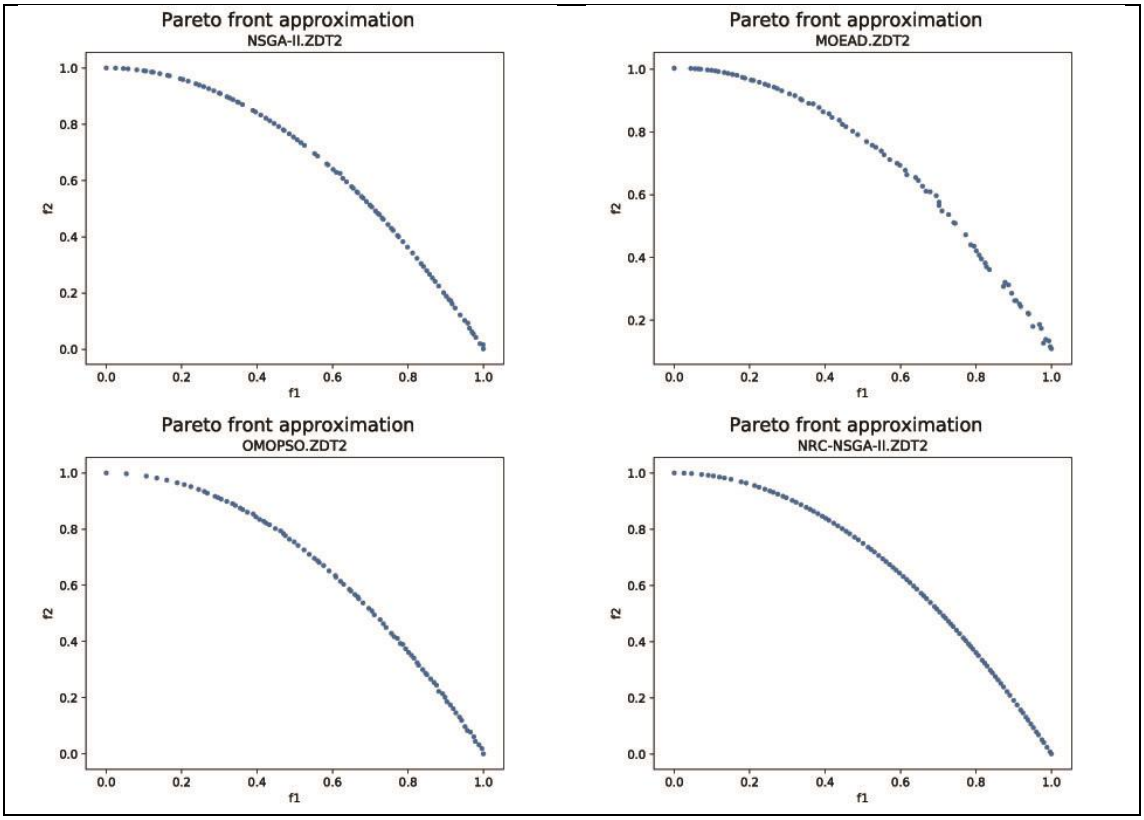

Figure 4. The Pareto fronts produced by four algorithms for ZDT2

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

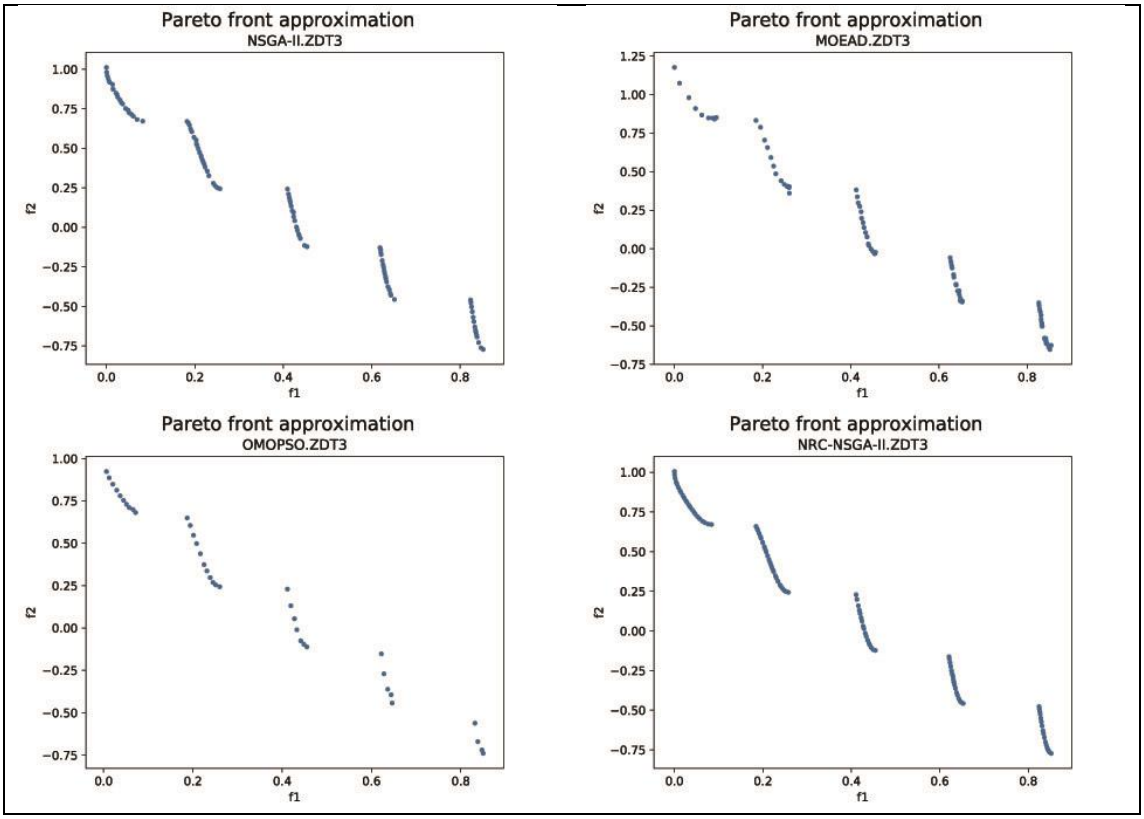

Figure 5. The Pareto fronts produced by four algorithms for ZDT3

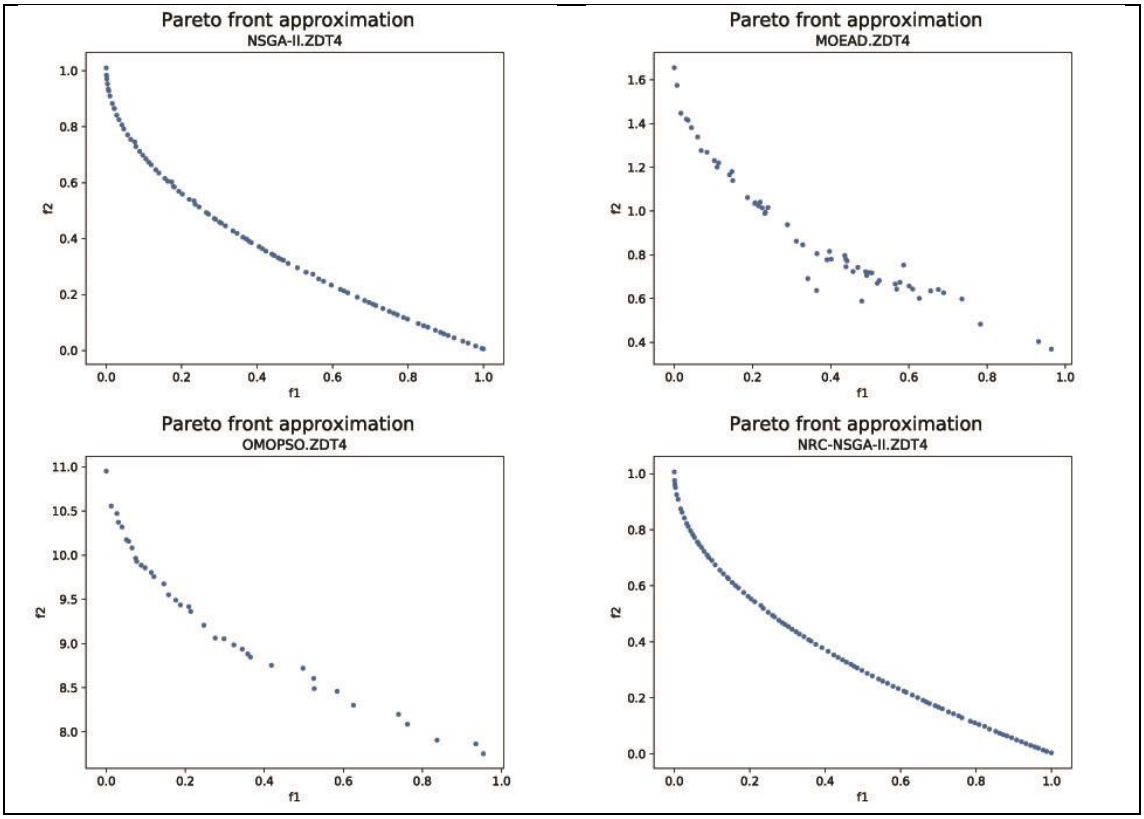

Figure 6. The Pareto fronts produced by four algorithms for ZDT4

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

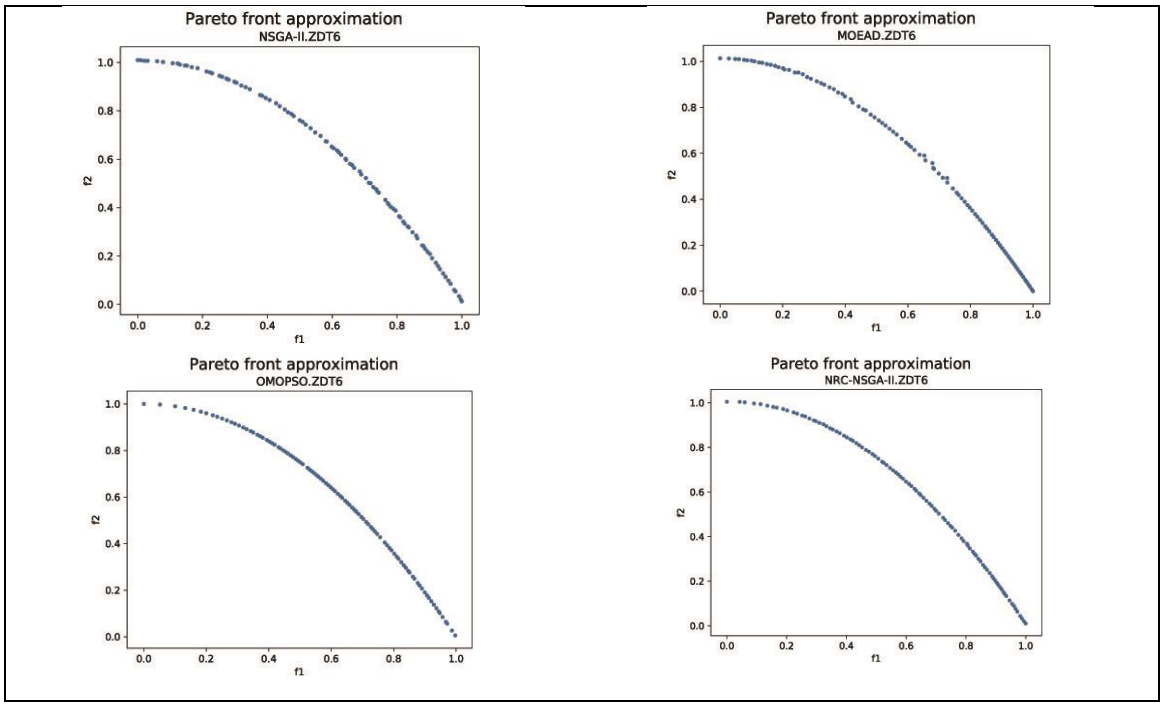

Figure 7. The Pareto fronts produced by four algorithms for ZDT6

Table 3. The performance metrics for ZDT series test functions

instance	Metric(GD)	NSGA-II	MOEA/D	OMOPSO	NRCNSGA-II
ZDT1	mean	0.001228	0.018069	2.764135	0.000725
	std	0.000528	0.006256	0.219743	7.05781e-05
ZDT2	mean	0.001285	0.031028	3.728351	0.000709
s	std	0.000632	0.014825	0.206830	0.000201
ZDT3	mean	0.001482	0.034499	2.585950	0.001237
	std	0.000610	0.012125	0.191269	0.000235
ZDT4	mean	0.249901	0.848660	114.3404	0.231035
	std	0.246983	0.577412	12.59204	0.200361
ZDT6	mean	0.056810	0.038294	6.974135	0.049901
	std	0.028173	0.001051	0.241325	0.024125

Table 4. The performance metrics for ZDT series test functions

instance	Metric(IGD)	NSGA-II	MOEA/D	OMOPSO	NRCNSGA-II
ZDT1	mean	0.004960	0.019316	2.280056	0.003870
	std	0.000159	0.006434	0.162155	4.35106e-05
ZDT2	mean	0.005089	0.032130	3.568274	0.004200
s	std	0.000174	0.016970	0.175149	0.000125
ZDT3	mean	0.005528	0.072778	1.951780	0.004535
	std	0.000227	0.023399	0.224967	9.58190e-05
ZDT4	mean	0.084858	0.696952	80.90478	0.107902
	std	0.101940	0.447998	11.84997	0.088262
ZDT6	mean	0.009900	0.004179	6.835721	0.006900
	std	0.000891	0.000752	0.277581	0.000693

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

It can be seen from the figure that the proposed algorithm is better than the other three algorithms. The Pareto solution set obtained by the NRCNSGA-II algorithm has better diversity. The GD metric can well confirm that NRCNSGA-II has good convergence, which proves that the non-uniform mutation operator we use is better than the polynomial mutation operator in solution set convergence. At the same time, according to the IGD metric, we confirmed that the proposed novel rank and crowded comparison operator can improve the diversity of Pareto solution set.

5. CONCLUSIONS

The convergence and diversity are two main goals in any MOEA. The crowding distance based method of NSGA-II fails to measure the solution diversity of the Pareto front in some cases. Therefore, this paper proposes a novel selection operator for optimizing the diversity of Pareto solution sets. At the same time, using the non-uniform mutation operator instead of the polynomial mutation operator to improve the convergence. The results show that the proposed algorithm NRCNSGA-II is effective.

Our future work will have two ideas: First, we will apply the proposed method to engineering applications. Second, we will will focus on optimizing higher-dimensional multi-objective problems.

ACKNOWLEDGMENTS

This work is supported by National key Research and Development Program of China under Grants nos. 2017YFB1103603 and 2017YFB1103003, National Natural Science Foundation of China under Grants nos. 61602343, 51607122, 61772365, 41772123, 61802280, 61806143, and 61502318, Tianjin Province Science and Technology Projects under Grants nos. 17JCYBJC15100 and 17JCQNJC04500, and Basic Scientific Research Business Funded Projects of Tianjin (2017KJ093, 2017KJ094).

REFERENCES

[1] K. Deb, A. Pratap, S. Agarwal, et al., A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactionson Evolutionary Computation. 6 (2) (2002) 182–197.

[2] E. Zitzler, M. Laumanns, and L. Thiele, SPEA2: improving the strength pareto evolutionary algorithm for multiobjective optimization,Evolutionary Methods for Design, Optimisation and Control, vol. 3242, pp. 95–100, CIMNE, Barcelona,Spain, 2002.

[3] Zhang Q, Li H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition[J]. IEEE Transactionson Evolutionary Computation, 2007, 11(6):712-731.

[4] M. Reyes and C. Coello. Improving pso-based multiobjective optimization using crowding, mutation and edominance. In Evolutionary Multi-Criterion Optimization (EMO 2005), pages 505–519, 2005.

[5] A.J. Nebro, J.J. Durillo, J. Garcia-Nieto, C.A. Coello Coello, et al., SMPSO: A new PSO-based metaheuristic for multi-objective optimization, IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM), 2009.

[6] K. Deb and N. Srinivas. Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation, pages 221–248, 1994.

[7] X.Q. Chen, Z.X. Hou, L.M. Guo, W.C. Luo, Improved multiobjective genetic algorithm based on NSGA-II, J. Comput. Appl.26(10) (2006), 2453–2456.

[8] L.I. Mi-Qing, Improved NSGA-2 algorithm based on minimum spanning tree, Comput. Eng. Appl. (2007), 170–179.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201911_5(11).0012

[9] L. Xuhong, L. Yushu, Z. Guoyin, Improvement of multi-objectiveoptimization algorithm NSGA-II, Comput. Eng. Appl. 41(15) (2015), 73–75. 1002-8331-(2005)15-0073-03

[10] K. Sindhya, K. Miettinen, and K. Deb. A hybrid framework for evolutionary multi-objective optimization. IEEE Transactions on Evolutionary Computation, 2012. in press.

[11] Xinchao Zhao,Convergent Analysis on Evolutionary Algorithm with Non-uniform Mutation, 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), 2008.

[12] Yijie S, Gongzhang S (2008) Improved NSGA-II multi-objective genetic algorithm based on hybridization-encouraged mechanism. Chin J Aeronaut 21(6):540–549

[13] Pourvaziri H, Naderi B (2014) A hybrid multi-population genetic algorithm for the dynamic facility layout problem. Appl Soft Comput 24:457–469

[14] Toledo CFM, Franca PM, Morabito R, Kimms A. Multi- population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem. Int J Prod Res 47(11):3097–3119, 2009.

[15] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, third ed., Springer, New York, 1996.

[16] M.R. Sierra, C.A.C. Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art, Int. J. Comput. Intell. Res. 2 (3) (2006) 287–308.

[17] A. Veldhuizen and G. Lamont. On measuring multiobjective evolutionary algorithm performance. 2000 Congress on Evolutionary Computation, pages 204–211, 2000.

[18] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. daFonseca, Performance assessment of multiobjective optimizers: An analysis and review, IEEE Trans. Evol. Comput., vol. 7, no. 2, pp. 117–132, Apr. 2003.

[19] E. Zitzler, K. Deb, and L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, Evol. Comput., vol. 8, no. 2, pp.173–195, 2000.