International Journal of Cognitive Informatics and Natural Intelligence
Volume 16 • Issue 1

A Many-Objective Practical Swarm Optimization Based on Mixture Uniform Design and Game Mechanism

Chen Yan, South China Agriculture University, China
Cai Mengxiang, South China Agriculture University, China*
Zheng Mingyong, South China Agriculture University, China
Li Kangshun, Guangdong University of Science and Technology, China

ABSTRACT

In recent years, multi-objective optimization algorithms, especially many-objective optimization algorithms, have developed rapidly and effectively. Among them, the algorithm based on particle swarm optimization has the characteristics of simple principle, few parameters, and easy implementation. However, these algorithms still have some shortcomings, but also face the problems of falling into the local optimal solution, slow convergence speed, and so on. In order to solve these problems, this paper proposes an algorithm called MUD-GMOPSO, a many-objective practical swarm optimization based on mixture uniform design and game mechanism. In this paper, the two improved methods are combined, and the convergence speed, accuracy, and robustness of the algorithm are greatly improved. In addition, the experimental results show that the algorithm has better performance than the four latest multi-objective or high-dimensional multi-objective optimization algorithms on three widely used benchmarks: DTLZ, WFG, and MAF.

KEYWORDS

Game Mechanism, Many-Objective Optimization, Mixture Uniform Design, Particle Swarm Optimization

1. INTRODUCTION

Multi-objective optimization problem(Kong et al., 2010) is one of the main forms of engineering practice and scientific research problems. There are often conflicts between objectives, and the optimal solution is a set of non dominant Pareto optimal solutions. Generally, when the number of objectives is 4 or more, it is called many-objective problem. As the number of conflicting objectives of many-objective problems increases, the number of non dominated solutions increases sharply, resulting in increased computational complexity and search difficulty. It is one of the most difficult problems in the field of intelligent optimization at home and abroad. In order to deal with this problem effectively, in recent years, many researchers have adopted different methods and technologies to solve the key problems in MaOPs from different angles. Here are four types of methods: (1) Pareto based methods;

DOI: 10.4018/IJCINI.301203 *Corresponding Author

(2) Decomposition based method; (3) Index based approach; (4) Particle swarm optimization based method. Among them, the method based on particle swarm optimization will be the focus of this paper.

1. **Pareto based methods:** Pareto dominance has always been the cornerstone of mainstream MOEAs and a hot research direction of scholars. Non-dominated Sorting Genetic Algorithm II, NSGA-II(Deb et al., 2002); Strength Pareto Evolutionary Algorithm II, SPEA-II(Zitzler et al., 2001); Pareto Envelope-Based Selection Algorithm II, PESA-II(Corne et al., 2001); etc. However, with the increase of the objective dimension of the optimization problem, the dominance relationship between individuals will become less and less obvious, and the selection pressure will drop sharply, resulting in the performance of the algorithm in high-dimensional optimization problems is far less excellent than that in low-dimensional optimization problems. In order to solve this problem, some scholars have made corresponding efforts in recent years. For example, a grid based evolutionary algorithm (GrEA)(Yang et al., 2013) introduces grid advantages and uses grid to help population convergence. In addition, some MOEAs adopt effective diversity maintenance strategies as the second option, such as the new shift based density estimation method (SDE)(Guerrero-Pena & Araujo, 2019) and NSGA-III(Deb & Jain, 2014), which is an enhanced version of NSGA-II to maintain diversity by using reference points.
2. **Decomposition based methods:** Decomposition based algorithm is an algorithm combining mathematical programming and evolutionary algorithm. The most widely used decomposition based algorithm is MOEA/D(Qingfu & Hui, 2007), which was originally designed to solve mops. However, due to its effectiveness in dealing with mops, a large number of decomposition based algorithms have been proposed to solve MaOPs in recent years(Ishibuchi et al., 2016). In addition, the decomposition based method also has disadvantages in dealing with MaOPs. In order to solve this problem, some methods based on weight vector adjustment have been studied recently. For example, the WS method is applied locally in MOEA/D-LWS(Liu et al., 2011), and the vertical distance between the solution and the weight vector in the target space is used in MOEA / D-DU(Huang & Liu, 2009).
3. **Index-based methods:** Hypervolume (HV)(Yang et al., 2019) as an index, it can measure the convergence and diversity of Pareto front approximation at the same time. However, with the increase of the number of targets, the computational complexity of calculating HV increases significantly, which may hinder the application of this algorithm in MaOPs. Although some studies have been proposed recently to improve the calculation speed of HV(While et al., 2014; Yen & He, 2014), they still cannot effectively deal with MaOPs with a large number of targets.
4. **Particle-Swarm-Optimization-based methods:** Particle swarm optimization algorithm is a population evolutionary algorithm. Its idea comes from artificial life and evolutionary computing theory. PSO algorithm comes from the research on the predation behavior of birds. Birds randomly search for food in the natural area. If there is only one piece of food in this area, the easiest way to find food is to search the surrounding area of the bird nearest to the food. When solving the optimization problem, the solution of the problem can be regarded as a particle without volume in the d-dimensional search space, which is equivalent to the position of the bird. Each particle has a fitness value determined by the optimization function and has its own position and velocity. It is an intelligent optimization algorithm based on group iterative search, which can process all objective functions in parallel. It has the advantages of simple operation and fast convergence. Its application to solving multi-objective problems has attracted extensive attention in academic circles(Cheng & Jin, 2015).

However, at present, most multi-objective particle swarm optimization mainly aims at the optimization problems of 2 or 3 objectives, that is, multi-objective particle swarm optimization (MOPSO) (A proposal for multiple objective particle swarm optimization, 2002). How to make particle swarm optimization effectively solve many-objective problems is still an urgent problem to

be solved. Especially when the dimension is too high (> 10), most algorithms still face the problems of loss of population diversity (Wickramasinghe et al., 2010), fast convergence and easy to fall into local optimization. In order to solve these problems, this paper proposes a many-objective practical swarm optimization based on mixture uniform design and game mechanism (MUD-GMOPSO), which is based on the adaptive scheme of Gaussian chaotic mutation and elite learning(Yu et al., 2020), as well as the mechanism and elite learning method using adaptive parameters, It provides a new idea for particle swarm optimization to solve many-objective optimization problems. The hybrid uniform design idea comes from the reference point-based strategy in NSGA-III algorithm. This strategy is introduced into the many-objective particle swarm optimization algorithm. Combined with the characteristics of particle swarm optimization, a new global optimal selection strategy and external file maintenance strategy are proposed, which can not only give full play to the advantages of rapid convergence of particle swarm optimization, but also maintain the diversity distribution of solution set. Compared with other typical many-objective algorithms, the results show that the proposed algorithm can maintain good convergence and set distribution in solving many-objective problems.

2. THE PROPOSED RESEARCH

2.1 Mixture Uniform Design

Most of the traditional multi-objective optimization algorithms based on decomposition use DAS and Dennis's method in the construction of reference points, but the disadvantages of this method are very obvious, and it can only be applied to the multi-objective optimization problems with lower dimensions. When the objective dimension is higher, there will be too many reference points. According to the formula, for example, when the dimension is 10 and each dimension is divided into 10 segments, there will be too many reference points, 92378 reference points will be generated. In NSGA-III(Deb & Jain, 2014), Deb and Jain's method, a new construction method, divides the construction of reference points into internal and external layers, reduces the score of objective function in each dimension, and greatly reduces the number of reference points. But whether it is DAS and Dennis' method or the improved DEB and Jain 'method, the number of reference points generated by these two construction methods depends on M and H values, that is, the score of objective function of each dimension of objective number. Although DEB and Jain's method has greatly reduced the number of reference points, when the target dimension exceeds 10, a large number of reference points will still be generated in the inner layer. The dense distribution of reference points in the step of individual contacting reference points will lead to the concentration distribution of individual solutions, which will affect the distribution of the algorithm. The number of reference points should be consistent with the number of individuals in the population. Therefore, this algorithm will adopt a more flexible construction method: hybrid uniform design. The number of reference points generated by this construction method does not depend on any parameters and is defined by the user himself. It can generate reference points which are exactly consistent with the number of individual solutions.

1. Generate a vector W_1, which is composed of all positive integers less than N and prime to each other. Let $W_2 = \left(1,2,\ldots,N\right)$, where N is the number of individuals.
2. The matrix W is obtained by using the following formula:

$$W = \mathrm{mod}\left(W_2^T W_1 - 1, N\right) + 1$$

3. Take any column M-1 from the matrix W to form a new matrix X_i. Calculate the value of CD_2 of the matrix X_i through the following formula, and select the X_i with the maximum

value of CD_2, where M is the target dimension. Let the matrix W have column K, $i = 1,2,\ldots,C_k^{M+1}$, then:

$$CD_2\left(x\right) = [(\frac{13}{12})^{M-1} - \frac{2^{2-M}}{N}\sum\nolimits_{i=1}^{N}\prod\nolimits_{j=1}^{M-1}\left(2 + \left|x_{ij} - 0.5\right| - \left|x_{ij} - 0.5\right|^2\right)$$

$$+\frac{1}{N}\sum\nolimits_{i,k=1}^{N}\prod\nolimits_{j=1}^{M-1}(1 + 0.5(|\, x_{ij} - 0.5\left| - \right| x_{kj} - 0.5\left| - \right| x_{ij} - x_{kj}))]^{\frac{1}{2}}$$

4. The set of reference points S can be obtained from the value X_{ij} of matrix X:

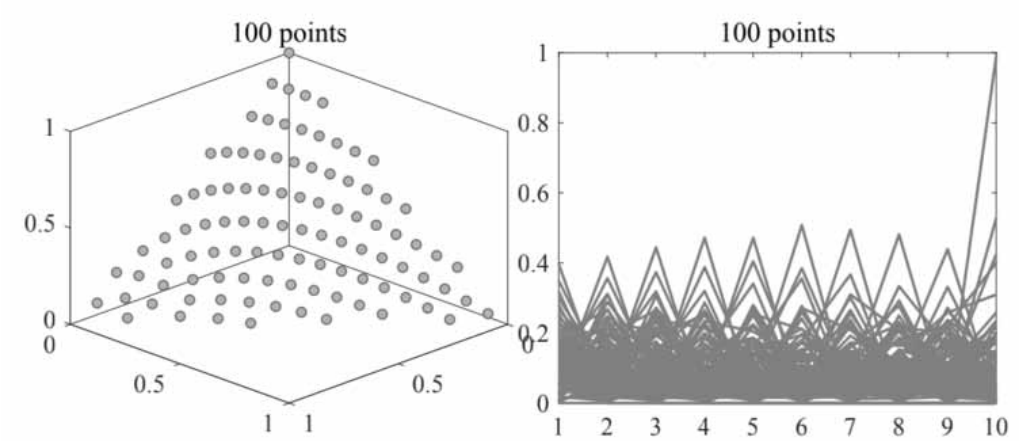
$$\begin{cases} s_{ij} = \dfrac{1}{N}\left[1 - \left(x_{ij}\right)^{\frac{1}{M-j}}\right]\prod\nolimits_{k=1}^{j-1}\left(x_{ik}\right)^{\frac{1}{M-k}} \ ,1 \le j < M \\ s_{ij} = \dfrac{1}{N}\prod\nolimits_{k=1}^{j-1}\left(x_{ik}\right)^{\frac{1}{M-k}} \ , j = M \end{cases}$$

Using the above 4 steps, the number of reference points is set to 100 for the target space with target dimensions of 3 and 10. The results are shown in Figure 1.

2.2 Improved Multi-Scale Mutation Strategy

The traditional multi-objective particle swarm optimization algorithm is famous for its fast convergence, so is the many-objective particle swarm optimization algorithm, but it is easy to fall into local optimum if it converges too fast. Although the mixed uniform design can effectively solve the problem of too many reference points, the problem of too fast convergence has not been well solved. A multi-scale mutation strategy was proposed in reference (Zhang et al., 2018), and good results were obtained in multi-objective problem. However, the core of this paper is the many-objective optimization problem. Therefore, this paper proposes an improved multi-scale mutation strategy based on reference (Zhang et al., 2018).

Figure 1. The reference points sampled by mixture uniform design on 3- and 10-objective unit simplexes



Algorithm 1 shows the multi-scale mutation strategy, which first uses crowding distance to arrange each particle in descending order. The top 20% of the particles are not densely distributed and are not easy to fall into local optimum, so small radius can be used μ* R, in which μ Is the variation scale (previously set as 0.3, in order to deal with higher dimensional problems, this paper set as 0.25); The last 20% particles are easy to fall into local optimum, so they use a larger variation radius R; In the middle 60% of the particles, because the probability of falling into the local optimum is between the two, so randomly choose one of the above two methods for mutation. See algorithm 1 for the specific mutation process. Where (m, n) is the chaotic sequence generated by the chaotic model, and crowding (POP) is to calculate the crowding distance of particles in the population and arrange them in descending order.

2.3 Game Mechanism

In the case of multi-objective, overemphasizing the fitness value often makes some particles far away from the local optimum, resulting in poor overall convergence effect of the algorithm, especially in the case of many-objective. Section 2.2 of this paper explains the improved variation scaling strategy. The game mechanism after adopting Section 2.2 strategy can well solve the problems of excessive selection pressure and particle easily falling into local optimum in many-objective optimization. The new game competition mechanism uses angle comparison instead of fitness comparison, which can effectively improve this kind of problem.

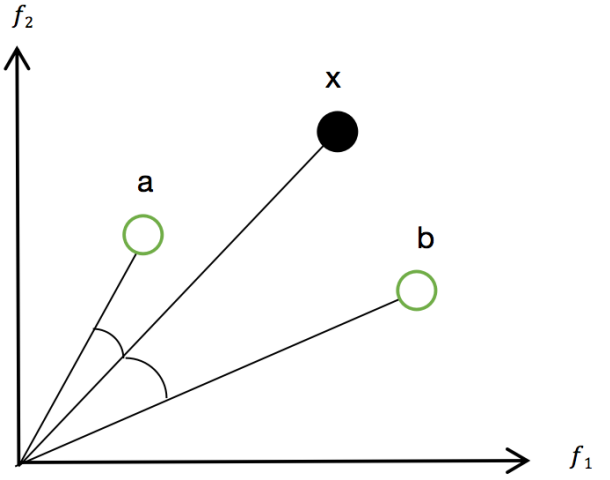
The new game update mechanism is shown in algorithm 2. The core process is that the non-elite particles randomly select two elite particles from the elite set and calculate the angle between the two elite particles and the non-elite particles. The particles with small angle will become the leader of the particles. The cycle of game mechanism begins with the determination of elite particles, which are selected by non-dominant sorting and crowding distance. Non dominant ranking is to stratify the population according to the level of non inferior solution. It is a cyclic fitness grading process. First, we find out the non-dominant solution of the first layer of the population, which is denoted as Figure 2. Then we delete the first layer, find the second layer, and so on, until all the particles are divided into different layers. Next, the crowding distance is calculated and arranged in descending order according to the particles in each layer. The crowding distance is the sum of the sides of the largest rectangle that can be formed between the particle and two adjacent particles in the same level.

The outstanding point of this mechanism is that elitist set particles only need to be selected by crowding distance and non-dominant sorting, and it is also very important for the selection of the size of elitist set. Smaller elitist set will lead to premature convergence, while larger elitist set can

Algorithm 1. New-Chaos-Mutation

Input: $\mu \ of \ pop\left(M,N\right)$
Output: Npop
1: r = ($X_{max} - X_{min}$) / 2, $\mu = 0.25$
2: Crowding(pop)
3: for I = 1:M
4: if (I < 0.2 * M) r = μ * r
5: else if (I > 0.8 * M) r = r
6: else if(rand > 0.5) r = μ * r
7: else r = r
8: end if
9: end if
10: Npop(I,:) = pop(I,:) + r * (2 * t - 1)
11: end for

Figure 2. The game of two elite particles



Algorithm 2. Game mechanism

Input: X,V,E,Iter,MaxIter
Output: NP
1: NP $\leftarrow \varnothing$
2: Crowding(pop)
3: for $X_i \in X$ do:
4: $X_a, X_b \leftarrow E$ (a,b$\in$ random)
5: θ_1 =angle(x,a), θ_2 =angle(x,b)
6: if $\theta_1 < \theta_2$ then $X_w = X_a$
7: else $X_a = X_w$
8: end if;
9: update X and V;
10: while Iter < 0.4*MaxIter :
11: NP $\leftarrow$ X,V
12: execute Algorithm 1
13: end for

delay the convergence speed and avoid falling into local optimum, so we set the size of the elite set to 10% of the population. When the elitist particles are determined, the game is carried out, and the winner of the game will guide other particles in the population to fly as the global optimal particles. In each pair of games, the particles to be updated randomly select two elitist particles A and B from the elitist set, and the two elitist particles play the game on the angle formed by the two and non-elitist particles from the far point of the coordinate axis, The game with small angle is successful. As shown in Figure 1, the angle between elite particle A and non elite particle x is small, so particle a guides particle x to update its speed and position. The speed update formula is as follows. The whole process from selecting elites to comparing is called game, because the selection of elites is random. It is not clear which leader will be selected by the particle to be updated. The properties of the leader

will determine the effect of particle update. The better the properties, the better the effect. On the contrary, the general properties of the leader will take the second place, The effect of particle update depends entirely on the leader, so it is called game.

The new velocity and position update formula of particles is as follows:

$$V_i^{'} = c_1 V_i + c_2 \left(X_w - X_i \right)$$
$$X_i^{'} = X_i + V_i^{'}$$

where c_1 and c_2 are randomly generated vectors between [0, 1], X is the position of the winner, Y is the current position of the particle, and Z is the current velocity of the particle.

2.4 General Framework

Based on the above three improvement strategies, the flow of MUD-GMOPSO proposed in this paper is as follows:

Step 1: Initialization. Set the population P (0), the population number N and the maximum number of iterations Gmax; Initialize the position X and velocity V of all particles; Set the file A(0) = ∅, and the maximum capacity B; Let t = 0.

Step 2: According to the hybrid design method, the reference points with the same number of particles are generated, and the target values of all non-dominated solutions are converted and associated with the corresponding reference points.

Step 3: Add the non-dominated solution in P (0) to A(0) to form A (1).

Step 4: Let t =t+1.

Step 5: According to the improved PSO formula of game mechanism, the velocity and position of particles are updated, and the objective function value is calculated.

Step 6: If the particle dominates the old solution in A (k), the particle is added to A(k) and the dominated old solution in A(k) is deleted. If the particles do not dominate each other, the number of dominated targets is determined first. If the number of targets is greater than 1/2 of the target dimension, the particles are added to A (k), and the dominated old solutions in A(k) are deleted.

Step 7: Judge whether the number of solutions in A(k) is greater than $\overline{\mathrm{A}}$. If so, delete the redundant solutions according to the external file maintenance strategy and the reference line p to which the solutions belong.

Step 8: If the algorithm termination condition is not reached, return to step 4; Otherwise, the algorithm is terminated and the non-dominated solution in A(k) is finally output.

The whole process is shown in Figure 3.

3. EXPERIMENTAL STUDY

3.1 Test Problem

The test function used in this paper is a popular series of test functions, which are DTLZ series, WFG series and MAF series. A total of 17 functions are used for multiple tests. DTLZ series is a kind of standard test functions composed of DTLZ1 to DTLZ7. WFG series is composed of WFG1-WFG5, MAF series is composed of MAF1-MAF5. DTLZ and WFG series standard test functions are tested with 5, 8, 10 and 15 targets, and the comparison algorithms are NSGA-III(Deb & Jain, 2014), CMOPSO(Zhang et al., 2018), VaEA(Xiang et al., 2017), MOEA/DD(Li et al., 2015), etc. these algorithms are commonly used classical comparison high-dimensional algorithms. NSGA-III

Figure 3. General framework of MUD-GMOPSO

is a traditional many-optimization algorithm based on reference point design; MOEA/DD combines the advantages of Pareto and decomposition; VaEA is an environment selection algorithm based on vector angle. Therefore, this paper will compare with these algorithms to verify the effectiveness of the algorithm.

3.2 Performance Measures

This experiment is completed in mac10.13.0 system. The performance evaluation indicator of the algorithm is IGD reverse generation distance and WV super volume. The population size of all algorithms is set to 100, and the archive capacity of the algorithm with external set is set to 100. The test function runs 300 times except DTLZ3 iteration for 1000 times. All algorithms run 50 times independently. The result is the average and standard deviation of IGD for 50 times. The smaller the IGD, the better the performance of the algorithm.

The IGD indicator measures the average distance between the final solution obtained by the algorithm and the reference point on the real Pareto front, and the calculation is as follows:

$$IGD\left(A,P^{*}\right)=\frac{\sum_{x\in P^{*}}\min_{\alpha\in A}\left(distance\left(x,\alpha\right)\right)}{\left|P^{*}\right|}$$

where A is the final approximate solution set obtained by the algorithm; It is the reference point on the real Pareto front; distance (x, A) is the Euclidean distance between X and point A. The biggest disadvantage of IGD is the need for reference points.

The HV indicator measures the size of the target space, which is determined by the final solution. Let z= $z_1, z_2, \ldots, z_m$, be the reference point dominated by all Pareto's best points. The calculation of HV is shown in formula:

$$HV\left(A,z\right) = Lebesgue\left(\bigcup_{x\in A}\left[f_1\left(x\right),z_1\right],\ldots,\left[f_m\left(x\right),z_m\right]\right)$$

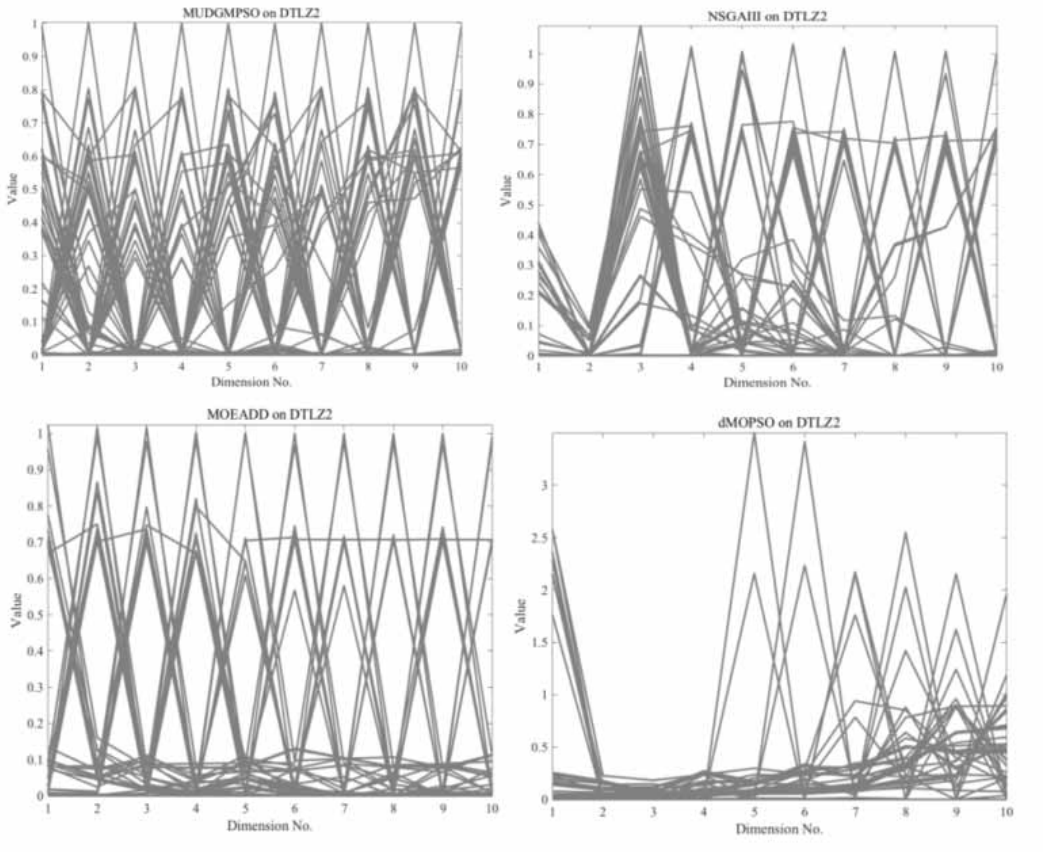
where Lebesgue (·) is a Lebesgue measure. Before calculating HV, the target values of the final solutions of all test problems are normalized with 1.1 × (fmax1, fmax2, fmaxm), where fmaxi (I Î 1, 2,..., m) is the maximum value of the ith objective of the actual Pareto front, then the reference point is set as [1, 1, 1]. In this work, when the number of objectives is not more than 8, the recently proposed method for calculating HV is used. In order to save computing resources, when the number of targets exceeds 8, Monte Carlo method is used to calculate the approximate value of HV.

3.3 Contrast Experiences

3.3.1 Comparison on DTLZ Series Functions

As shown in Figure 4, MUD-GMOPSO is compared with other common many-objective optimization algorithms on DTLZ2 function in Pareto frontier. It can be seen from the figure that through 10000

Figure 4. Pareto front of each algorithms on DTLZ2 function



iterations, the algorithm is closer to Pareto frontier than other algorithms, which shows that the algorithm has strong convergence on DTLZ2 and is not easy to fall into local optimum.

For the problems from DTLZ1 to DTLZ4, MUD-GMOPSO outperforms other algorithms in IGD in 10 items, and outperforms other algorithms in HV in 6 items. Taking NSGA-III as an example, the performance of MUD-GMOPSO is better than that of NSGA-III in high-dimensional target space when IGD value is used as the performance index. In this experiment using HV value as evaluation index, the difference between the two algorithms is very small no matter in which dimension. The reason is that the individual solutions obtained by the evolutionary algorithm based on the reference point in the target space are guided by the preset reference point, and the approximate solutions close to the reference point do not necessarily maximize the super volume. The calculation method of IGD value proposed by DEB can be more specific for the characteristics of this kind of algorithm and reflect the performance difference of the algorithm.

MUD-GMOPSO has a good effect on DTLZ1 in low dimension (5-8), and on DTLZ2, 3 and 4 in high dimension (10-15). It can be seen that MUD-GMOPSO can play a greater advantage in high dimension for most DTLZ algorithms. The data set is shown in Table 1.

Table 1. Comparison of IGD and HV optimal values of each function on DTLZ functions

	m	MaxGen	MUD-GMOPSO	NSGA-III	CMOPSO	MOEA/DD
DTLZ1	5	600	**3.265×10^{-2}** **5.123×10^{-2}**	4.236×10^{-2} 5.329×10^{-2}	4.215×10^{-2} 9.031×10^{-2}	5.178×10^{-2} 8.343×10^{-2}
	8	750	**1.781×10^{-2}** 8.712×10^{-2}	3.141×10^{-2} **6.462×10^{-2}**	1.981×10^{-2} 5.862×10^{-2}	3.145×10^{-2} 8.312×10^{-2}
	10	1000	**2.099×10^{-2}** 4.743×10^{-2}	2.246×10^{-2} 4.214×10^{-2}	2.446×10^{-2} 4.322×10^{-2}	2.235×10^{-2} **3.874×10^{-2}**
	15	1500	3.152×10^{-2} 1.437×10^{-1}	**2.649×10^{-2}** 1.101×10^{-1}	3.032×10^{-2} 7.478×10^{-1}	4.562×10^{-2} **1.110×10^{-1}**
DTLZ2	5	350	2.953×10^{-2} 2.965×10^{-2}	3.944×10^{-2} 5.347×10^{-2}	**9.944×10^{-3}** **2.347×10^{-2}**	3.853×10^{-1} 2.998×10^{-1}
	8	500	**3.378×10^{-2}** 8.117×10^{-2}	4.365×10^{-2} 7.769×10^{-2}	2.315×10^{-2} 5.179×10^{-2}	4.589×10^{-1} **6.606×10^{-1}**
	10	750	**3.128×10^{-2}** 7.902×10^{-2}	6.149×10^{-2} 7.365×10^{-2}	4.123×10^{-2} **5.402×10^{-2}**	7.504×10^{-1} 7.928×10^{-1}
	15	1000	**3.964×10^{-2}** **6.083×10^{-2}**	4.593×10^{-2} 9.014×10^{-2}	5.954×10^{-2} 9.014×10^{-2}	9.024×10^{-1} 1.066×10^{-1}
DTLZ3	5	350	**8.013×10^{-3}** 2.911×10^{-2}	8.656×10^{-3} **2.763×10^{-2}**	9.653×10^{-3} 2.853×10^{-2}	8.487×10^{-2} 1.115×10^{-1}
	8	500	9.342×10^{-3} 7.880×10^{-2}	8.769×10^{-3} **5.564×10^{-2}**	**8.656×10^{-3}** 2.673×10^{-2}	2.659×10^{-1} 3.211×10^{-1}
	10	750	**9.881×10^{-3}** **2.849×10^{-2}**	1.126×10^{-2} 3.559×10^{-2}	8.169×10^{-3} 5.468×10^{-2}	5.985×10^{-1} 6.880×10^{-1}
	15	1000	1.617×10^{-2} **4.211×10^{-2}**	**1.502×10^{-2}** 4.721×10^{-2}	9.881×10^{-3} 3.559×10^{-2}	7.174×10^{-1} 8.526×10^{-1}
DTLZ4	5	350	**1.046×10^{-2}** 4.691×10^{-2}	1.789×10^{-2} **4.001×10^{-2}**	1.458×10^{-1} 5.805×10^{-1}	2.418×10^{-1} 4.421×10^{-1}
	8	500	2.413×10^{-3} 5.878×10^{-1}	**2.044×10^{-3}** **5.272×10^{-1}**	1.459×10^{-1} 4.321×10^{-1}	3.446×10^{-1} 6.391×10^{-1}
	10	750	**1.110×10^{-2}** **4.369×10^{-2}**	2.694×10^{-2} 6.340×10^{-2}	9.984×10^{-2} 9.773×10^{-1}	6.676×10^{-1} 9.543×10^{-1}
	15	1000	**3.078×10^{-2}** **9.174×10^{-2}**	3.159×10^{-2} 1.073×10^{0}	2.684×10^{-1} 1.066×10^{0}	8.486×10^{-1} 1.239×10^{0}

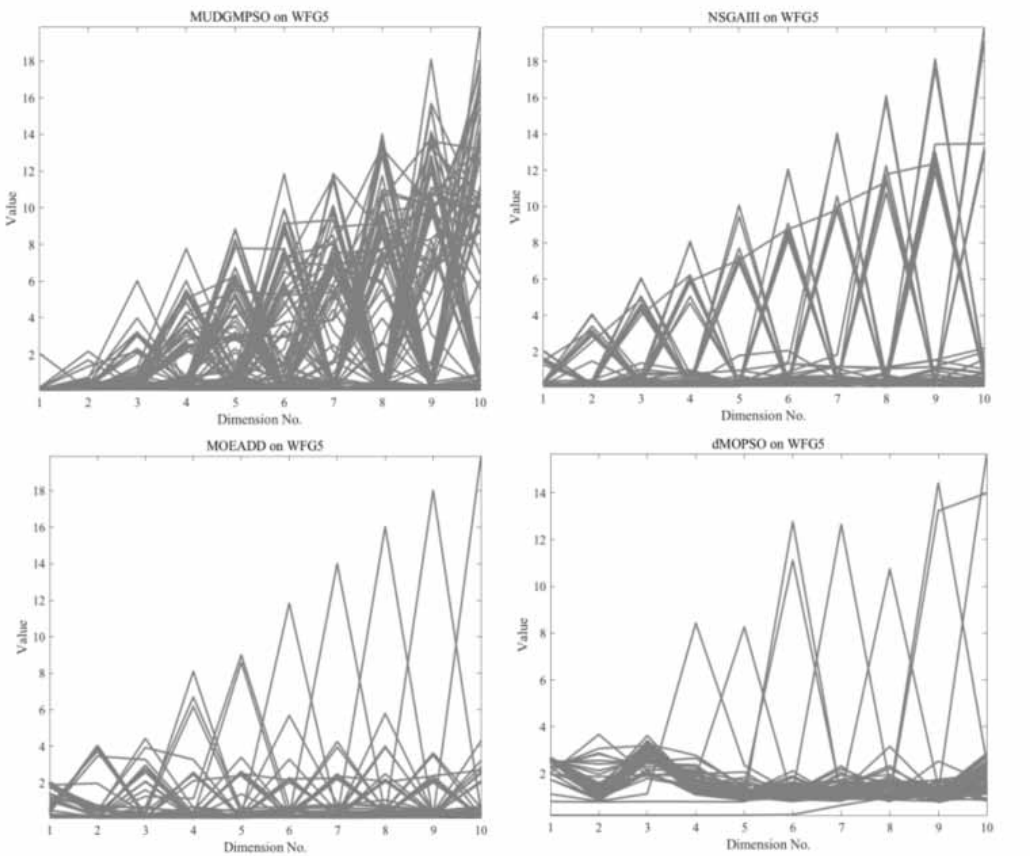
3.3.2 Comparison on WFG Series Functions

As shown in Figure 5, MUD-GMOPSO is compared with other common many-objective optimization algorithms in the Pareto frontier on WFG5 function. It can be seen from the figure that through 10000 iterations, the algorithm is closer to the Pareto front than other algorithms, which shows that the algorithm also has strong convergence on WFG5 and is not easy to fall into local optimum.

In WFG1 and WFG2, MUD-GMOPSO has average performance in low dimensional target space, but better performance in high dimensional target space. Experiments show that MUD-GMOPSO has a relatively good performance in 5-10 dimensional WFG2, but when the target space dimension rises to 15 dimensional, the algorithm performance has a serious decline. By observing the test problem DTLZ7 with similar geometry to WFG2 and WFG1, it is found that when the target space dimension rises to 15, the algorithm performance will decline obviously. WFG2 and DTLZ7 decreased significantly. The reason is that the reference points constructed in the high-dimensional space can keep the distribution of individual solutions, but cannot guide the individual solutions to converge to the irregular Pareto front, so the convergence performance of the algorithm is poor. It shows that MUD-GMOPSO has insufficient ability to deal with mixed and separated test problems.

In addition, MUD-GMOPSO performs better in WFG3. In all dimensions of the target space, the performance is in the front, and compared with NSGA-III and other algorithms, there is a big gap. The characteristics of WFG3 and WFG6 are similar. They are both unimodal and indecomposable optimization problems. The difference between them is that the geometry of WFG6 is concave, while

Figure 5. Pareto front of each algorithm on MaF4 function



WFG3 is linear and degenerate. Therefore, it is speculated that MUD-GMOPSO is more suitable for solving optimization problems with degenerate geometry than other algorithms.

In general, for WFG1 to WFG4, MUD-GMOPSO performs better in IGD than other algorithms in 9 data items, and in HV than other algorithms in 7 data items. Specifically, MUD-GMOPSO has a similar advantage in low and high dimensions for WFG1, 2, 3 and 4, but for other algorithms, especially NSGA-III algorithm, each index is still dominant. It can be seen that MUD-GMOPSO can play a greater advantage in low and high dimensions for most WFG algorithms. The data set is shown in Table 2.

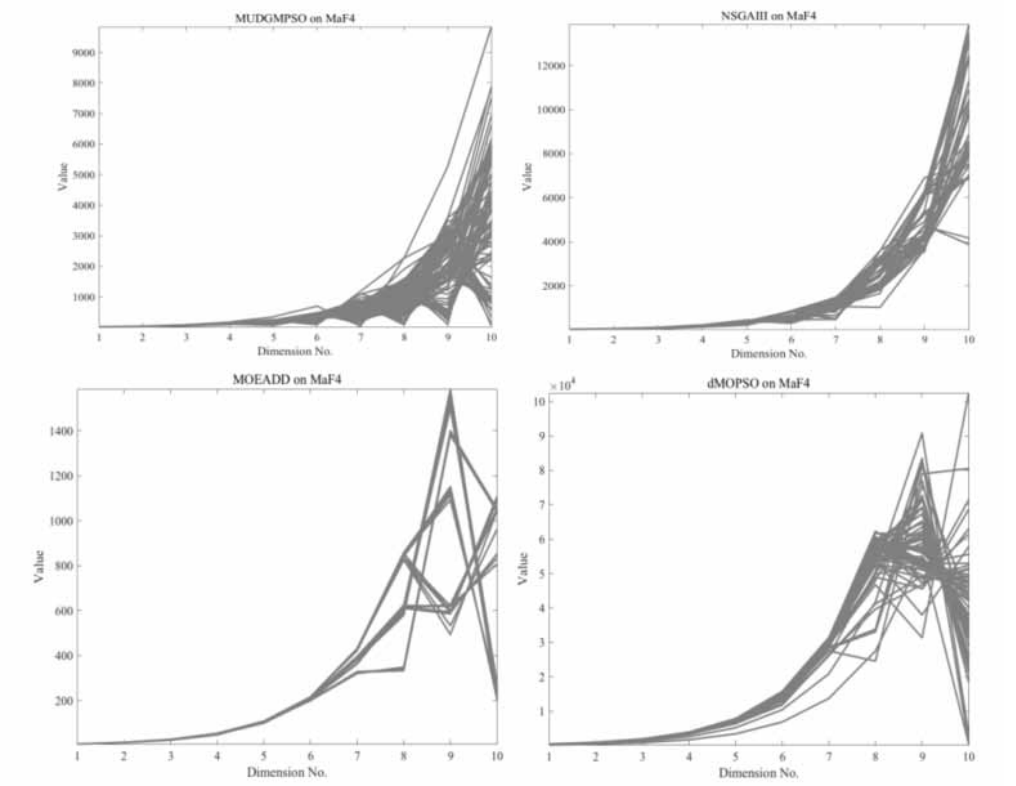
3.3.3 Comparison on MaF Series Functions

As shown in Figure 6, MUD-GMOPSO is compared with other common many-objective optimization algorithms in the Pareto frontier on MaF4 function. It can be seen from the figure that through

Table 2.

	m	MaxGen	MUD-GMOPSO	NSGA-III	CMOPSO	MOEA/DD
WFG1	5	600	3.211×10^{-2} **4.980×10^{-2}**	4.659×10^{-2} 6.901×10^{-2}	**2.757×10^{-2}** 5.122×10^{-2}	1.007×10^{-2} 5.259×10^{-2}
	8	750	2.315×10^{-2} **6.206×10^{-2}**	**1.917×10^{-2}** 8.178×10^{-2}	2.645×10^{-2} 6.252×10^{-2}	4.492×10^{-2} 5.556×10^{-2}
	10	1000	**2.277×10^{-2}** **5.770×10^{-2}**	4.019×10^{-2} 8.571×10^{-2}	2.899×10^{-2} 6.870×10^{-2}	2.419×10^{-2} 8.547×10^{-2}
	15	1500	2.585×10^{-2} **6.343×10^{-2}**	4.203×10^{-2} 9.007×10^{-2}	**2.406×10^{-2}** 6.815×10^{-2}	1.700×10^{-2} 5.371×10^{-2}
WFG2	5	350	**2.127×10^{-2}** 5.306×10^{-2}	2.924×10^{-2} 6.773×10^{-2}	2.566×10^{-2} **6.603×10^{-2}**	1.007×10^{-2} 5.259×10^{-2}
	8	500	**3.106×10^{-2}** **4.781×10^{-2}**	4.366×10^{-2} 6.994×10^{-2}	3.625×10^{-2} 5.841×10^{-2}	1.917×10^{-2} 5.556×10^{-2}
	10	750	**3.198×10^{-2}** 6.524×10^{-2}	4.842×10^{-2} 8.134×10^{-2}	3.689×10^{-2} **8.243×10^{-2}**	2.419×10^{-2} 8.547×10^{-2}
	15	1000	2.700×10^{-2} 6.232×10^{-2}	4.575×10^{-2} 9.571×10^{-2}	2.321×10^{-2} 6.345×10^{-2}	3.870×10^{-2} **5.371×10^{-2}**
WFG3	5	350	3.357×10^{-2} **5.711×10^{-2}**	4.003×10^{-2} 6.221×10^{-2}	**3.218×10^{-2}** 6.632×10^{-2}	**3.218×10^{-2}** 6.632×10^{-2}
	8	500	**3.581×10^{-2}** **5.100×10^{-2}**	4.931×10^{-2} 7.415×10^{-2}	3.713×10^{-2} 5.110×10^{-2}	5.463×10^{-2} 5.709×10^{-2}
	10	750	**2.299×10^{-2}** 7.335×10^{-2}	4.910×10^{-2} 7.759×10^{-2}	2.988×10^{-2} 6.554×10^{-2}	3.796×10^{-2} **5.410×10^{-2}**
	15	1000	**3.713×10^{-2}** 4.826×10^{-2}	5.201×10^{-2} **3.918×10^{-2}**	3.837×10^{-2} 4.543×10^{-2}	6.212×10^{-2} 5.534×10^{-2}
WFG4	5	350	3.257×10^{-2} **4.767×10^{-2}**	5.231×10^{-3} 6.754×10^{-2}	**2.867×10^{-2}** 5.042×10^{-2}	**3.353×10^{-2}** 5.042×10^{-2}
	8	500	**3.287×10^{-2}** 5.692×10^{-2}	5.344×10^{-2} **5.430×10^{-2}**	3.581×10^{-2} 7.302×10^{-2}	4.254×10^{-2} **4.598×10^{-2}**
	10	750	**3.331×10^{-2}** 7.325×10^{-2}	6.048×10^{-2} 8.163×10^{-2}	3.569×10^{-2} **5.268×10^{-2}**	4.121×10^{-2} 6.256×10^{-2}
	15	1000	3.119×10^{-2} **6.606×10^{-2}**	6.919×10^{-2} 8.278×10^{-2}	**2.149×10^{-2}** 8.577×10^{-2}	5.051×10^{-2} 9.412×10^{-2}

Figure 6. Pareto front of each algorithm on MaF4 function



10000 iterations, the algorithm is closer to Pareto frontier than other algorithms, which shows that the algorithm has strong convergence on MaF4 and is not easy to fall into local optimum. But for DTLZ2 and WFG5 functions, the effect is worse.

MaF1 is the inversion function of DTLZ1, NSGA-III performs best in HV, and MUD-GMOPSO performs better in IGD. Among them, MUD-GMOPSO has the best performance in the case of 15 targets, because its IGD value is the smallest; MaF2 is improved on the basis of DTLZ2, which increases the difficulty of convergence. In low dimensional space, CMOPSO performs best in IGD and HV, but MUD-GMOPSO still dominates in 10-15 targets; For MaF3, it has a convex PF and a large number of local PF. The dominant effect of MUD-GMOPSO is worse than that of MaF1 and MaF2, which indicates that MUD-GMOPSO has some shortcomings in dealing with functions with a large number of local PF.

For MaF1 to MaF4 problems, in general, MUD-GMOPSO outperforms other algorithms in terms of IGD in 13 items, and outperforms other algorithms in terms of HV in 7 items. Specifically, MUD-GMOPSO has a better effect on maf4 in the low dimension (5-8), and has a better effect on MaF1 and MaF2 in the high dimension (10-15). Although its effect on MaF3 is poor, it is not different from other algorithms. It can be seen that MUD-GMOPSO can play a greater advantage in high dimension for most MaF algorithms. The data set is shown in Table 3.

CONCLUSION

With the rapid development of economy and the continuous progress of society, more and more factors need to be considered in the optimization problems in production and life. The traditional

Table 3.

	m	MaxGen	MUD-GMOPSO	NSGA-III	CMOPSO	MOEA/DD
MaF1	5	600	**1.046×10^{-2}** 4.691×10^{-2}	4.236×10^{-2} **3.329×10^{-2}**	4.215×10^{-2} 9.031×10^{-2}	5.178×10^{-2} 8.343×10^{-2}
	8	750	**2.413×10^{-3}** 5.272×10^{-2}	3.141×10^{-2} **4.962×10^{-2}**	1.981×10^{-2} 5.862×10^{-2}	3.145×10^{-2} 8.312×10^{-2}
	10	1000	**1.110×10^{-2}** 4.369×10^{-2}	2.246×10^{-2} **3.874×10^{-2}**	2.446×10^{-2} 4.322×10^{-2}	2.235×10^{-2} 4.234×10^{-2}
	15	1500	**2.078×10^{-2}** **9.438×10^{-2}**	3.649×10^{-2} 1.101×10^{-1}	3.032×10^{-2} 7.478×10^{-1}	4.562×10^{-2} 1.110×10^{-1}
MaF2	5	350	2.953×10^{-2} 2.965×10^{-2}	3.944×10^{-2} 5.347×10^{-2}	**9.944×10^{-3}** **2.347×10^{-2}**	3.853×10^{-1} 2.998×10^{-1}
	8	500	**3.378×10^{-2}** 8.117×10^{-2}	4.365×10^{-2} 7.769×10^{-2}	2.315×10^{-2} 5.179×10^{-2}	4.589×10^{-1} **6.606×10^{-1}**
	10	750	**3.128×10^{-2}** 7.902×10^{-2}	6.149×10^{-2} 7.365×10^{-2}	4.123×10^{-2} **5.402×10^{-2}**	7.504×10^{-1} 7.928×10^{-1}
	15	1000	**3.964×10^{-2}** **7.231×10^{-2}**	4.593×10^{-2} 9.014×10^{-2}	5.954×10^{-2} 9.014×10^{-2}	9.024×10^{-1} 1.066×10^{-1}
MaF3	5	350	**8.013×10^{-3}** 2.911×10^{-2}	8.656×10^{-3} **2.763×10^{-2}**	9.653×10^{-3} 2.853×10^{-2}	3.265×10^{-2} 5.123×10^{-2}
	8	500	**8.656×10^{-3}** 7.880×10^{-2}	8.769×10^{-3} 5.564×10^{-2}	**9.342×10^{-3}** 8.512×10^{-2}	1.781×10^{-2} **2.673×10^{-2}**
	10	750	**9.881×10^{-3}** **2.849×10^{-2}**	1.126×10^{-2} 3.559×10^{-2}	8.169×10^{-3} 5.468×10^{-2}	2.099×10^{-2} 4.743×10^{-2}
	15	1000	1.617×10^{-2} **4.211×10^{-2}**	**1.502×10^{-2}** 4.721×10^{-2}	9.881×10^{-3} 3.559×10^{-2}	3.152×10^{-2} 1.437×10^{-1}
MaF4	5	350	**3.265×10^{-2}** **4.001×10^{-2}**	1.789×10^{-2} **4.001×10^{-2}**	1.458×10^{-1} 5.805×10^{-1}	2.418×10^{-1} 4.421×10^{-1}
	8	500	**1.781×10^{-2}** 8.712×10^{-2}	2.044×10^{-2} 5.878×10^{-2}	1.459×10^{-1} 6.754×10^{-2}	3.446×10^{-1} **4.321×10^{-2}**
	10	750	**2.099×10^{-2}** **4.743×10^{-2}**	2.694×10^{-2} 6.340×10^{-2}	9.984×10^{-2} 9.773×10^{-1}	6.676×10^{-1} 9.543×10^{-1}
	15	1000	3.152×10^{-2} 1.437×10^{-1}	3.159×10^{-2} 1.073×10^{-1}	**2.684×10^{-2}** **1.066×10^{-1}**	8.486×10^{-2} 1.239×10^{-1}

multi-objective optimization algorithm is not suitable for solving these problems in production and life. Therefore, this paper proposes an algorithm called Many-Objective Practical Swarm Optimization based on Mixture Uniform Design and Game mechanism (MUD-GMOPSO). In the algorithm, a relatively low computational complexity method is used to screen and update the particles in each generation through hybrid uniform design and game between particles. At the same time, a new leader selection scheme in particle swarm optimization algorithm is proposed. This paper mainly carried out the following work:

1. To solve the problem that the reference point construction method used in NSGA-III needs to rely on the target dimension and the score of each target direction, a hybrid uniform design method is introduced. The reference points generated by the hybrid uniform design do not depend on any parameters, and the number of reference points is defined by the user. This method can keep

the individual solutions of the population more evenly distributed in different target directions in the evolution process, and ensure the distribution of the algorithm.

2. Aiming at the problem that the traditional many-objective optimization algorithm has insufficient convergence in the high-dimensional target space, a new game competition combination mechanism is designed based on the idea of game and competition between particles in literature (Yu et al., 2020) and (Zhang et al., 2018). At the same time, the NSGA-III individual solution is used to contact the reference vector to maintain the original excellent distribution of the algorithm. At the same time, the distance from the individual solution to the reference vector and the distance from the projection point of the individual solution on the reference vector to the ideal point are comprehensively considered. After the ranking evaluation is calculated, the individual solution is sorted and screened. The experimental results show that this mechanism can ensure the distribution of the algorithm and enhance the convergence of the algorithm in high-dimensional space. In most cases, the performance of the algorithm is better than NSGA-III, MOEA/DD and other classic many-objective optimization algorithms.
3. In order to verify the effectiveness of MUD-GMOPSO algorithm, a series of experiments are carried out on WFG, DTLZ and MaF problems, and compared with NSGA-III(Deb & Jain, 2014), CMOPSO(Zhang et al., 2018), VaEA(Xiang et al., 2017), MOEA/DD(Li et al., 2015) and other algorithms. The results show that MUD-GMOPSO algorithm has better performance than other four algorithms.

To sum up, the improved algorithm proposed in this paper successfully achieves the purpose of improving the performance of the algorithm, and explores the factors that may affect the performance of the algorithm, which provides ideas for the follow-up research (Li & Chen, 2018; Li & Liang, 2019; Li & Wang, 2019; Wang, Li, Liao et al, 2020; Wang, Li, Zhou et al, 2020). In the future, I will continue to explore the algorithm from different angles, such as combining with other algorithms to further improve the performance of the algorithm, and extend the algorithm to solve irregular PF problems or real problems.

ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation of Guangdong Province of China with the Grant No.2020A1515010784 and the Key R&D Program of Guangdong Province with No. 2019B020219003.

REFERENCES

A proposal for multiple objective particle swarm optimization. (2002). *Proceedings of the IEEE Congress on Evolutionary Computation*, 1051-1056.

Cheng, , RJin, , Y. (2015). A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, *291*(10), 43–60.

Corne, D. W., Jerram, N. R., Knowles, J. D., & Oates, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, 283-290.

Deb, K., & Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, *18*(4), 577–601.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Guerrero-Pena, E., & Araujo, A. F. R. (2019). Multi-objective evolutionary algorithm with prediction in the objective space. *Information Sciences*, *501*, 293–316.

Huang, X., & Liu, B. (2009). Chaotic Mutation Particle Swarm Optimization based on Fuzzy Theory. *Journal of Fuzhou University*, *37*(05), 652–656.

Ishibuchi, H., Setoguchi, Y., Masuda, H., & Nojima, Y. (2016). Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. *IEEE Transactions on Evolutionary Computation*, *21*(2), 169–190.

Kong, W. J., Ding, J. L., & Chai, T. Y. (2010). Survey on large-dimensional multi-objective evolutionary algorithms. *Control and Decision*, *25*(3), 321–326.

Li, K., & Chen, Y. (2018). Improved gene expression programming to solve the inverse problem for ordinary differential equations. *Swarm and Evolutionary Computation*, *38*, 231–239.

Li, K., Deb, K., Zhang, Q., & Kwong, S. (2015). An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, *19*(5), 694–716.

Li, K., & Liang, Z. (2019). Performance Analyses of Differential Evolution Algorithm Based on Dynamic Fitness Landscape. *International Journal of Cognitive Informatics and Natural Intelligence*, *13*(1), 36–61.

Li, K., & Wang, H. (2019). A mobile node localization algorithm based on an overlapping self-adjustment mechanism. *Information Sciences*, *481*, 635–649.

Liu, Y., Ben, L., & Zhao, Q. (2011). Clustering Multi-objective Particle Swarm Optimization Algorithm based on Uniform Design. *Computer Engineering*, *37*(14), 152–154.

Qingfu, Z., & Hui, L. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, *11*(6), 712–731.

Wang, F., Li, Y., Liao, F., & Yan, H. (2020). An ensemble learning based prediction strategy for dynamic multi-objective optimization. *Applied Soft Computing*. Advance online publication. doi:10.1016/j.asoc.2020.106592

Wang, F., Li, Y., Zhou, A., & Tang, K. (2020). An Estimation of Distribution Algorithm for Mixed-Variable Newsvendor Problems. *IEEE Transactions on Evolutionary Computation*, *24*(3), 479–493.

While, L., Bradstreet, L., & Barone, L. (2014). A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, *18*(4), 481–502.

Wickramasinghe, U. K., Carrese, R., & Li, X. (2010). Designing airfoils using a reference point based evolutionary many-objective particle swarm optimization algorithm. In *Proc of IEEE Congress on Evolutionary Computation*. IEEE.

Xiang, Y., Zhou, Y., Li, M., & Chen, Z. (2017). A vector angle-based evolutionary algorithm for unconstrained many-objective optimization. *IEEE Transactions on Evolutionary Computation*, *21*(1), 131–152.

Yang, S., Li, M., Liu, X., & Zheng, J. (2013). A grid-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, *17*(5), 721–736.

Yang, T., Tang, D., & Cao, S. (2019). Many-objective Particle Swarm Optimization Algorithm based on Hyperspherical Fuzzy Domination. *Computer Applications (Nottingham)*, *39*(11), 3233–3241.

Yen, G. G., & He, Z. (2014). Performance metric ensemble for multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *18*(1), 131–144.

Yu, J., Wang, W., Wu, G., & Liang, W. (2020). Game mechanism based multi-objective particle swarm optimization. *Computer Engineering and Design*, *41*(04), 964–971.

Zhang, X., Zheng, X., Cheng, R., Qiu, J., & Jin, Y. (2018). A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. *Information Sciences*, *427*, 63–76.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *Proceedings of the Fifth Conference on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, 95-100.