An improved evolutionary multi-objective optimization algorithm based on decomposition

Jiaxin Han*, Manman He, Ying Cuan
School of Computer Science, Xi’an Shiyou University, Xi’an, Shaanxi, 710065
*Corresponding author’s e-mail: jxhan@xsyu.edu.cn

Abstract: Multi-objective optimization (MOP) is a hot and difficult problem in the field of industrial production and scientific research, involving simultaneous optimization of multiple objectives. In this paper, an improved decomposition-based multi-objective evolutionary algorithm is proposed to solve MOP. Firstly, the Latin hypercube sampling method is used to generate the initial population in order to improve the diversity of the initial population and the probability of convergence to the global optimal solution. Secondly, the adaptive differential crossover operator is used to adjust the parameter settings adaptively for different problems, including the differential proportionality factor and crossover probability, so as to improve the performance of the algorithm. Finally, the reference point is set to coordinate origin, which can pull the population back to the real front faster and accelerate the convergence speed of the population. In a group of benchmark problems, ZDT and DTLZ test problems, the algorithm is widely compared with four most advanced algorithms. The experimental results show that the algorithm has better ability to solve MOP problems.

1. Introduction
In recent years, multi-objective optimization algorithms have developed very rapidly, and have practical application in many engineering fields. This type of problem has many objective functions that need to be optimized. At the same time, these objective functions are conflicted each other. The promotion of one objective function will certainly lead to the deterioration of other objective functions. Since the goal of a multi-objective optimization algorithm is to obtain a non-dominated solution set rather than a single solution, it is still a difficult problem for dealing with MOPs.

The traditional approach for solving multi-objective problems is to translate multi-objective problems into single-objective problems, including weighted sum method, principal objective method and so on. The shortcoming of these methods is that some parameters need to be determined empirically.

Evolutionary algorithm optimizes a group of individuals at the same time, and can generate multiple approximate solutions in the optimization process. The contemporary evolutionary multi-objective optimization algorithm [1] introduces new evolution paradigms, such as multi-objective particle swarm optimization [2], multi-objective optimization based on artificial immune system [3], multi-objective optimization based on distributed estimation algorithm [4] and evolutionary multi-objective optimization algorithm based on decomposition [5]. In 2007, Zhang et al. [5] proposed a decomposition-based multi-objective evolutionary algorithm (MOEA/D). MOEA/D decomposes the multi-objective problem into a group of single-objective problems by decomposition strategy, and optimizes the group of single-objective problems simultaneously. Among most of MOEA/D, the differential scale factor and cross probability of differential crossover operator are set according to empiricism, however this type of parameter setting is not always the best for different problems.
In this paper, we propose an improved MOEA/D based on Latin hypercube sampling, self-adaptive differential crossover operator and set the origin as the reference point, the proposed is denoted as MOEA/D-LH-SA, which could adaptively adjust parameter settings. The main contributions of MOEA/D-LH-SA is presented as follows:

- Latin hypercube sampling is used to generate initial population to enhance the diversity of the initial population;
- A self-adaptive differential crossover operator is used to replace the simulated binary crossover operator of MOEA/D to achieve that adaptively adjusting parameter settings including differential scale factor and cross probability for different problems.
- Setting the origin as the reference point for use is to allow the population to converge faster to the real frontier.

The rest of this paper is presented as follows. Section II presents the proposed algorithm in detail. Section III provides the experimental results and analysis. The last Section makes a conclusion.

2. The Proposed Algorithm

In the following, we describe in detail Latin hypercube sampling and self-adaptive differential crossover operator, before the overall flow of MOEA/D-LH-SA is presented.

2.1. Latin Hypercube Sampling

As we all known, population initialization is very important in genetic algorithms, it affects the global convergence speed of the algorithm and the quality of the solution. In the absence of any historical information available, the general algorithm uses a random initialization method. However, this method cannot guarantee that the initial population is widely distributed in the whole problem space, which affects the efficiency of algorithm execution in a way.

In this paper, the Latin hypercube sampling [6] is used to generate initial population to enhance the diversity of the initial population. Supposing that the dimension of the variable is n, \( x_i \in [x_i^l, x_i^u] \), \( i = 1, 2, ..., n \), \( x_i \) is the i-th dimensional variables, \( x_i^l \) and \( x_i^u \) are the upper and lower bounds of \( x_i \), respectively. Then the description of Latin hypercube sampling is shown in Algorithm 1.

### Algorithm 1: Latin Hypercube Sampling

**Input:** population size N  
**Output:** the initial population

**Step 1:** Divide the interval \([x_i^l, x_i^u] \) of each \( x_i \) into N equal subintervals,

\[
x_i^l = x_i^0 < x_i^1 < x_i^2 < x_i^3 < x_i^4 < x_i^5 \cdots x_i^N = x_i^u;
\]

**Step 2:** Generate a matrix A with the size of \( N \times n \), each column of A is a random full permutation of the sequences \{1, 2, 3, ..., N\}:

**Step 3:** Each row of A corresponds to a selected small hypercube, and a sample is randomly generated in each selected small hypercube, so that N samples can be obtained.

2.2. Self-adaptive Difference Crossover Operator

Since the setting of the differential scale factor (\( F \)) and the crossover probability (\( CR \)) in the differential evolution operator [7] is problem-dependent, so we proposed a self-adaptive differential crossover operator which could adaptively adjust parameter settings of F and CR for different problems.

In self-adaptive differential crossover operator, \( F \) and \( CR \) are evolved along with the population individual. The initial values of \( F \) and \( CR \) are 0.5 and 0.9, respectively. The main motivation of self-adaptive differential crossover operator is to save the number of \( F \) and \( CR \) that can produce better individuals to the next generation with greater probability. The description of adaptively adjusting \( F \) and \( CR \) of i-th individual is shown as follows.

\[
oldCR_i = CR, oldF_i = F
\]
\[
newCR_i = \begin{cases} 
0.0 + 1.0 \times \text{rand}[0,1] & \text{if } \text{rand}[0,1] < 0.1 \\
oldCR_i & \text{otherwise}
\end{cases} 
\]

\[
newF_i = \begin{cases} 
0.1 + 0.9 \times \text{rand}[0,1] & \text{if } \text{rand}[0,1] < 0.1 \\
oldF_i & \text{otherwise}
\end{cases} 
\]

2.3. The Setting of Reference Point

The reference point reflects the change trend of Pareto frontier. Of course, if the change trend is closer to the origin, the Pareto frontier can be quickly pulled back to the real Pareto frontier, so that the population converges to the real Pareto frontier more quickly. Therefore, this paper sets zero vectors whose reference points are consistent with the dimension of the objective function.

2.4. The Framework of MOEA/D-LH-SA

MOEA/D-LH-SA use Latin hypercube sampling to generate initial population, adopt a self-adaptive differential crossover operator as the crossover operator and set the origin as the reference point. The framework of MOEA/D-LH-SA is introduced in Algorithm 2.

**Algorithm 2: MOEA/D-LH-SA**

**Input:** N (population size), \( T \) (the number of the weight vectors in the neighborhood of each weight vector)

**Output:** the final population

**Step 1:** Initialization:

**Step 1.1:** Generate N uniformly distributed weight vectors \( \lambda^1, \lambda^2, \ldots, \lambda^N \);

**Step 1.2:** For each weight vector \( \lambda^i \), calculating the Euclidean distance between \( \lambda^i \) and other weight vectors, and find the T closest weight vectors of \( \lambda^i \). The T weight vectors are the neighborhood of \( \lambda^i \). For any \( i = 1, \ldots, N \), there are \( B(i) = (i, \ldots, i) \), \( \lambda^1, \ldots, \lambda^N \) are the closest T weight vectors;

**Step 1.3:** Use the Latin hypercube sampling to generate initial population \( x^1, x^2, \ldots, x^N \) and calculating the corresponding objective function value \( FV^i = F(x^i) \);

**Step 1.4:** Initialization reference point m dimension \( z = (0, \ldots, 0)^T \);

**Step 2:** Update: for \( i = 1, \ldots, N \)

**Step 2.1:** Genetic operation: randomly select three indexes \( r_1, r_2, r_3 \) from \( B(i) \), perform adaptive differential crossover operator on individuals \( x^{r_1}, x^{r_2}, x^{r_3} \) generate a new solution \( y' \), and perform polynomial mutation operations on the new solution \( y' \), generating a new solution \( y \);

**Step 2.2:** Fix the new solution: If a dimension of \( y \) exceeds the search boundary, correcting the new solution \( y \) according to the specific problem to ensure each dimension of the new solution is within the search scope;

**Step 2.3:** Update the neighborhood solutions: For any \( j \in B(i) \), if \( g(y|x^i, z) \leq g(x^j|x^i, z) \), then set \( x^j = y \) and \( FV^j = F(y) \);

**Step 3:** Determine the termination condition:
If the termination condition is met, the algorithm stops and outputs \( \{x^1, \ldots, x^N\} \) and \( \{F(x^1), \ldots, F(x^N)\} \), otherwise, go to Step 2.

3. Experimental Studies

This section verifies the performance of MOEA/D-LH-SA by conducting a comparison between MOEA/D-LH-SA and other two effective multi-objective optimization algorithms, Non-dominated
Sorting Genetic Algorithm-II (NSGA-II) [8], and MOEA/D [5]. This section mainly introduced the benchmark functions, parameter settings of all the algorithms, performance metrics and experimental results and analysis.

3.1. Benchmark Functions
In this section, ZDT [9] series functions including ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 and DTLZ [10] series functions including DTLZ1, DTLZ2, DTLZ3, DTLZ4 and DTLZ6 are used to validate the performance of the proposed algorithm. Each algorithm is performed 10 independent runs on each benchmark function.

3.2. Parameter Settings
Most parameters in these comparative algorithms are set according to their original references, Where CR and F are the control parameters of the differential evolution operator, CR is the crossover probability and sets 0.9, F is the scaling factor and sets 0.5, $\eta_m$ and $p_m$ are the distribution index and mutation probability of the polynomial mutation, respectively setting 20 and $1/n$ ($n$ denotes the number of decision variables). $\eta_c$ and $p_c$ are the distribution indices and crossover probabilities of the simulated binary crossover operator, respectively setting 20 and 1. $T$ represents the size of neighborhood of each weight and sets 20 in MOEA/D, $N$ is population size, for 2-objective problems, the population size is 100 and the population size is 210 for 3-objective problems.

3.3. Performance Metrics
Here, reverse generation distance (IGD) [11] is used to evaluate the performance the compared algorithms, which could evaluate both convergence and diversity.

Assuming that $P^*$ is the true Pareto front, $P$ is the Pareto front found by an algorithm, IGD is defined as follows:

$$ IGDP^*,P) = \sum_{v \in P^*} \frac{d(v,P')}{|P^*|} $$

(4)

Where $d(v,P') = \min_{u \in P'} ||F(v) - F(u)||$ is the distance between individual $v$ and its nearest neighbor in $P'$, and $|P^*|$ is the cardinality of $P^*$. The smaller the IGD value, the better the performance the algorithm.

3.4. Experimental Results and Analysis
Since these algorithms are close to the front-end and real-front of test functions ZDT1, ZDT2, ZDT3 and ZDT6, this paper mainly lists images which are quite different from real-front, such as test functions ZDT4, DTLZ1, DTLZ2 and DTLZ3. The approximation between the Pareto frontier and the ideal frontier is measured experimentally in figure1.-figure4 on ZDT and DTLZ test family. Red indicates the test value and black represents the ideal Pareto frontier.

Figure 1. Plots of ZDT4 in NSGA-II, MOEA/D and MOEA/D-LH-SA experimental Pareto values and true Pareto Frontiers on 2-objective test instance.
In this paper, we compare the MOEA/D-LH-SA with the existing two evolutionary multi-objective optimization algorithms, NSGA-II and MOEA/D. The mean and variance of the IGD values of the three algorithms are shown in table 1. And all the results are the mean and variance of 10 independent runs.

It can be seen from table 1 that MOEA/D-LH-SA outperforms the other two algorithms in the test problems ZDT2, ZDT4, ZDT6, DTLZ2 and DTLZ3. For ZDT1, NSGA-II is the best algorithm, MOEA/D-LH-SA performs similar to NSGA-II. Unfortunately, MOEA/D-LH-SA performs worse on ZDT3.

In a word, the proposed MOEA/D-LH-SA algorithm is superior to other comparison algorithms on most benchmark functions.

4. Conclusion
This paper presents an improved MOEA/D, i.e. MOEA/D-LH-SA, which use Latin hypercube sampling to generate initial population to enhance the diversity of the initial population, use a self-adaptive differential crossover operator to adaptively adjust parameter settings of differential scale factor and cross probability for different problems and using origin as reference point, the population can approach the real frontier more quickly.
The comparison between MOEA/D-LH-SA and the existing two evolutionary multi-objective optimization algorithms indicates the better performance of the proposed algorithm. Further research could apply the proposed algorithm to solve complex high-dimensional multi-objective optimization problems.

Table 1. Experimental results of different algorithms.

| Functions | statistics | NSGA-II | MOEA/D | MOEA/D-LH-SA |
|-----------|------------|---------|--------|--------------|
| ZDT1      | mean       | 0.0043  | 0.0048 | 0.0044       |
|           | variance   | 3.72E-04| 2.64E-04| 1.28E-04    |
| ZDT2      | mean       | 0.0043  | 0.0041 | 0.0040       |
|           | variance   | 1.72E-04| 8.33E-05| 3.56E-05    |
| ZDT3      | mean       | 0.0046  | 0.0165 | 0.1492       |
|           | variance   | 2.11E-04| 0.0120 | 0.0052       |
| ZDT4      | mean       | 0.0098  | 0.0050 | 0.0044       |
|           | variance   | 0.0173  | 0.0010 | 3.22E-04     |
| ZDT6      | mean       | 0.0023  | 0.0019 | 0.0018       |
|           | variance   | 0.3241  | 1.74E-05| 7.91E-06    |
| DTLZ1     | mean       | 0.1216  | 0.0570 | 0.0827       |
|           | variance   | 0.1397  | 0.0901 | 0.1305       |
| DTLZ2     | mean       | 0.0491  | 0.0454 | 0.0453       |
|           | variance   | 0.0031  | 3.39E-04| 3.69E-04    |
| DTLZ3     | mean       | 0.1656  | 0.0508 | 0.0496       |
|           | variance   | 0.3648  | 0.0021 | 0.0014       |

Acknowledgement
Our thanks to the Shaanxi Provincial International Co-operation and Exchanges in Science and Technology Plan Project (2019kw-045) and Shaanxi Provincial Natural Science Foundation (2018JM6093).

Reference
[1] Gong M G, Jiao L C, Yang D D, Ma W P. Research on evolutionary multi-object optimization algorithm. Journal of software 2009, 20(2):271-289.
[2] Coello C A C and Lechuga M S. MOPSO: a proposal for multiple objective particle swarm optimization. Proceedings of Congress on Evolutionary Computation, Honolulu, USA, May 2002, 2: 1051-1056.
[3] Gong M G, Jiao L C and Du H F. Multiobjective immune algorithm with nondominated neighbor-based selection. Evolutionary computation, 2008, 16(2): 225-255.
[4] Zhang Q F, Zhou A M and Jin Y. RM-MEDA: A regularity model based multi-objective estimation if distribution algorithm. IEEE Transactions on Evolutionary Computation, 2007, 12(1): 41-63.
[5] Zhang Q F and Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731.
[6] Iman R L. Latin Hypercube Sampling. Encyclopedia of Quantitative Risk Analysis and Assessment. 2008.
[7] Price K, Storn R M, Lampinen J A. Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series), 2005.
[8] Deb K, Pratap A, Agarwal S and Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans on Evolutionary Computation, 2002, 6(2): 182-197.
[9] Zitzler E, Deb K, Thiele L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 2000, 8(2):173-195.
[10] Deb K, Thiele L, Laumanns M, et al. Scalable multi-objective optimization test problems. Congress on Evolutionary Computation. IEEE, 2002.

[11] Deb K, Jain H. 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. 2014, 18(4): 577-601.