# Learning to Accelerate Evolutionary Search for Large-Scale Multiobjective Optimization

Songbai Liu, Jun Li, Qiuzhen Lin, Member, IEEE, Ye Tian, and Kay Chen Tan, Fellow, IEEE

**Abstract**

This paper proposes an adaptive large-scale multi-objective evolutionary algorithm, in which a neural network-based accelerating optimizer is designed in the first stage to speed up the population's convergence and a competitive swarm optimizer is used in the second stage to maintain the population's diversity by spreading the solutions obtained in the first stage. To properly train the neural network in the first stage, the whole population, i.e., the training data, is evenly divided into two subsets with different qualities based on the dominant relationship between solutions. Then, the paired low-quality solutions and high-quality solutions, respectively, act as the input and the expected output when training the neural network. In this way, the potentially directional improvement information of the evolutionary population can be learned by this neural network, which is used to guide the adopted differential evolution in promising search directions. Once the population is detected to be evolutionarily stagnated in the first stage, the second stage will be activated for remedying the population's diversity. Experimental studies validate the performance of the post-innovation algorithm evolutionary large-scale optimizer when compared with pre-innovation algorithm in large-scale multi-objective optimization problems with decision variables ranging from 100 to 1000.

**Keywords**: Evolutionary multitasking, feature selec tion (FS), high-dimensional classification, particle swarm optimization (PSO).

## 1 Introduction

Multi-objective optimization problems (MOPs) usually contain several conflicting objectives that need to be optimized simultaneously[1] , as defined by

$$\begin{aligned} &\text{minimize } \ F(x) = (f_1(x), \ldots, f_m(x)), \\ &\text{subject to } \ x \in \Omega \end{aligned} \tag{1}$$

where $x = (x_1, ..., x_n)$ denotes the *n*-dimensional decision vector of a solution from the search space $\Omega$ and $F(x)$ defines *m* objective functions. Due to the conflicts often arising in different objectives, there is no a single optimal solution, but a set of equally optimal solutions termed Pareto-optimal set (PS) for solving MOPs. The mapping of PS onto the objective space is termed Pareto-optimal front (PF)[2]. Particularly, the problem in (1) is called a large-scale multi-objective optimization problem (LMOP) when the number of decision variables *n* is no less than 100[3]. During the past few decades, a number of multi-objective evolutionary algorithms (MOEAs) have been proposed with very effective performance for tackling MOPs[4-8]. However, the experimental results show that most existing MOEAs become not so efficient when solving LMOPs with a large-scale decision space, due to their weak search abilities[9-10]. To better solve LMOPs, a number of large-scale MOEAs

(LMOEAs) have been designed and most of them can be roughly divided into three categories[11], i.e., LMOEAs based on decision variable grouping strategies[12-13], based on decision space reduction techniques[14-16], and based on efficient search operators[17-18].

The first kind of LMOEAs applies different variable grouping strategies to divide the decision variables into several groups and then optimizes each group independently. For example, interdependence variable analysis and control variable analysis methods were designed in MOEA/DVA[12] to classify the decision variables into position variables, distance variables and mixed variables, which are then used to fine-tune the convergence and diversity of evolutionary population. In order to divide the decision variables more generically, an angle-based clustering method was proposed in LMEA[13] to group the decision variables into convergence-related and diversity-related variables. Then, these two different types of variables are optimized alternately by using different search strategies.

The second category of LMOEAs converts LMOPs into small-scale problems by using some problem transformation or dimension reduction techniques. Then, traditional MOEAs can be directly used to search the optimal solutions in a smaller search space. For example, a weighted optimization framework was designed in WOF[14], which tries to optimize the weight vector for the best objective value of each solution. Then, an efficient framework via problem reformulation was proposed in LSMOF[15], which applies the direction vectors and weight variables to reduce the dimensions of the target LMOP.

Different from the above two types of LMOEAs that reduce the difficulty or dimension of the target LMOP, the last kind of LMOEAs designs more efficient search strategies or operators to directly solve LMOPs, which can provide stronger search abilities[17-18]. Particularly, competitive swarm optimizer (CSO), as an improved variant of particle swarm optimizer (PSO), is one of the representative methods of this kind of LMOEAs, which shows its competitiveness in solving LMOPs due to the efficient search capability and inherent adaptability[19]. Some related works of CSOs for solving LMOPs will be introduced in detail in Section 2.2.

Given above, this paper proposes an adaptive large-scale multi-objective evolutionary algorithm, called AEA, for solving LMOPs. Specifically, the main contributions of this paper are clarified as follows:

- An adaptive two-stage strategy is proposed in this paper, where the first and second stages focus more on speeding up convergence and enhancing diversity of the evolutionary population, respectively. Once the population is detected to be evolutionarily stagnated in the first stage, the second stage will be activated for remedying the population's diversity. In this way, AEA can fully realize the promising performance complementarity between accelerating evolutionary search and maintaining population diversity when solving LMOPs with different difficulties.
- An accelerating optimizer is designed in the first stage of AEA to speed up the population's convergence. To be specific, an artificial neural network (ANN) is trained by using paired low-quality solutions and high-quality solutions, respectively, acting as the input and the expected output in the prepared training data.
- A CSO is designed in the second stage of AEA, aiming to remedy the population's diversity

obtained in the first stage.

## 2 Related works

### 2.1 Artificial neural network

In 1986, *Rumelhart* and *Meclelland et.al.* proposed a new learning procedure based on the back-propagating error for networks of neuron-like units[20], known as back-propagation neural network (BPNN), which is used to repeatedly adjust the weights of the difference between actual output vectors and desired output vectors of the network. The main purpose of this model training is to find a set of weights to ensure that the actual output vectors produced by the network are the same as (or infinitely close to) the desired output vectors.

Generally, the BPNN model training consists of two stages, including a ***Feed-Forward*** process and a ***Back-Propagation*** process, respectively. In the first stage, the input samples are fed into the input layer, and then transmitted to the output layer after being processed layer by layer in several hidden layers. After that, the actual output vectors generated by BPNN model are used to compare with the desired output vectors, thus the current total error *E* of the model is calculated by a loss function and then is used to compare with the predefined desired minimum error $E_{min}$. If and only if the current total error *E* is no longer larger than $E_{min}$, the model training would be finished. Otherwise, model training will enter the second stage (i.e., the ***back-propagation*** process), which applies the current total error *E* to repeatedly adjust the related parameters of neural net (i.e., weight vectors and biases) via gradient descent.

### 2.2 Competitive swarm optimizer

In 2015, *Cheng* and *Jin* first proposed a competitive swarm optimizer (CSO), which introduces the concept of competition mechanism between particles within one single swarm[19]. Different from traditional PSOs[21-23] that apply the global best and personal best particles to guide the search process of the swarm, CSO introduces a pairwise competition mechanism to achieve the swarm evolution. To be specific, the swarm is first divided into two groups, including the winner particles and loser particles. Then, the velocity and position of loser particles are updated by learning from their corresponding winner particles[19], formulated as follows:

$$\begin{aligned} V_l(t+1) = r_1 V_l(t) + r_2 \left(X_w(t) - X_l(t)\right) \\ +\varphi r_3 \left(\bar{X}(t) - X_l(t)\right) \end{aligned}, \tag{2}$$

$$X_l(t+1) = X_l(t) + V_l(t+1), \tag{3}$$

where $r_1$, $r_2$ and $r_3$ are three real-valued vectors randomly generated in (0, 1), $\bar{V}(t)$ and $\bar{X}(t)$ mean the average velocity and position of the relevant particles at $t^{th}$ iteration, respectively. $\varphi$ indicates a corresponding control parameter.

Empirical results have verified the effectiveness of CSOs in solving single-objective optimization problems with large-scale decision variables (LSOPs), such as two-phase based learning swarm optimizer (TPLSO)[24], population entropy based swarm optimizer (DCSO)[25], and level-based learning swarm optimizer[26]. Recently, CSOs have been successfully extended to solve LMOPs and shown efficiency. For instance, to improve the

search efficiency, a two-stage strategy was suggested in LMOCSO[27] to update the position of particles, which first preupdates the position of each particle based on its previous velocity, and then updates the position of each preupdated particle by learning from a leader particle. A tri-competition mechanism was proposed in S-ECSO[28], which can achieve a good trade-off between exploration and exploitation. A strong convex sparse operator was also implemented in S-ECSO, aiming to generate sparse particles during the position update process. Moreover, comprehensive competitive learning (CCL) strategy was proposed in CCSO[29], which uses three competition mechanisms, including environmental, cognitive and social competitions, to guide the particle search.

## 3 Method

This section provides a brief introduction of neural network-based accelerating optimizer.

With the increase of decision variables, LMOEAs need more computational resources to tackle the LMOPs, which is caused by the poor efficiency of their adopted search operators (e.g., DE, SBX and PSO). Therefore, an accelerating optimizer includes three main procedures: 1) training a neural network model $M_{NN}$ with three layers as shown in Fig. 1; 2) designing an improved DE operator based on the learned $M_{NN}$ to reproduce an offspring population by searching along more promising directions, and 3) running the Pareto-based environmental selection to get an updated population for the next generation. As described in Section 2.1, the neural network has an excellent capability of learning from the training data. In this paper, the used $M_{NN}$ tries to learn the potentially directional improvement information of the population, which can be employed to drive the population converging faster. Here, an improved DE operator driven by the learned $M_{NN}$ is proposed to generate a child solution $x^c$ starting from the parent solution $x$, as follows:

$$x^c = x + t(x^{P_1} - x^{P_2}) + (1-t)(x^{hq} - x) \tag{4}$$

where $x^{hq}$ is obtained by inputting $x$ into the $M_{NN}$, and $x^{P_1}$ and $x^{P_2}$ are two randomly selected solutions from the current population *P*. This formula mainly consists of two parts: the distribution part (i.e., $x^{P_1} - x^{P_2}$) and the acceleration part (i.e., $x^{hq}$-$x$).

Algorithm 1 presents the whole process of the neural network-based accelerating optimizer, which requires to input the current population *P*. In particular, the offspring population *Q* is initialized as an empty set. Then, the current population is equally divided into two subsets (i.e., $S_l$ represents the low-quality subset and $S_h$ represents the high-quality subset) by the nondominated sorting[30] in line 2. To properly train the initialized $M_{NN}$ with random parameters in line 3, the solutions in $S_l$ are iteratively inputted into this $M_{NN}$, correspondingly, the solutions in $S_h$ are acted as the expected output in training $M_{NN}$. Thus, both the input layer and the output layer have n nodes, which equal to the number of decision variables for the target LMOP. Besides, for each input solution in $S_l$, a random solution in $S_h$ is selected as its expected output. By learning from the progress of low-quality solutions moving toward high-quality solutions, the potentially directional improvement information of the evolutionary population can be extracted in this learned $M_{NN}$. Starting from

the position of a solution $x$, the $x^{hq}$ obtained via input $x$ into the $M_{NN}$ in line 5 can be used to guide the DE operator searching along more promising directions. In addition, two different solutions are randomly selected as parents in line 6. By this way, a new child solution $x^c$ with higher quality can be produced in line 7 by the improved DE operator, and the $x^c$ is saved in the offspring population $Q$ in line 8. Finally, the accelerating optimizer selects more promising solutions to survive to the next generation by a Pareto-based environmental selection, which runs a non-dominated sorting first, followed by selecting solutions with better dominant ranks and crowding distances, as suggested in[30].

**Algorithm 1** Accelerating Optimizer($P$

1: Initialize $Q$ as an empty set;
2: Divide $P$ into two layers $S_l$ and $S_h$ via non-dominated sorting;
3: $M_{NN}$ = Training ($S_l$, $S_h$);
4: **for** each $x \in P$ do **do**
5: obtain a new solution $x^{hq}$ by inputting $x$ to the $M_{NN}$;
6: select two different solutions ($x^{P_1}$, $x^{P_2}$) as parents;
7: reproduce a child $x^c$ with solutions ($x^{hq}$, $x^{P_1}$, $x^{P_2}$) by (4);
8: add $x^c$ into $Q$;
9: **end for**
10: $P$ = Pareto-Based Environmental Selection($P$, $Q$);
11: **return** $P$;

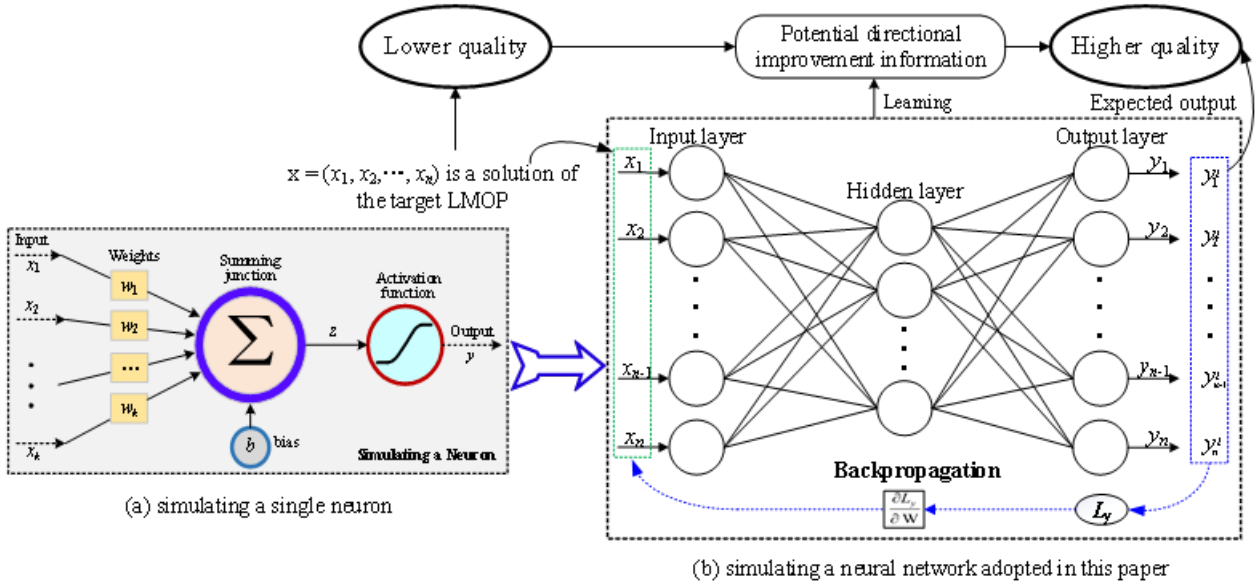



Figure 1: Illustration of the general components in a neuron and the constructed neural network.

## 4 Implementation details

### 4.1 Comparing with released source codes

No related source codes are available to my work. So here are the details of my own work, creative additions, noticeable improvements and/or new features. In addition, I do some improvements. On the bias of the accelerating optimizer, I embed the classical CSO to maintain the diversity of the population. Besides, I design a survival rate to indicate the quality of current population, which is used to adjust the proportion of accelerating optimizer and CSO. Therefore, the proposed algorithm is called AEA. The original algorithm is called AMOEA/D.

### 4.2 Experimental environment setup

We use the platform named Matlab R2020b for the implementation of the algorithm. To study the performance of AEA, it is compared with original and pre-innovation AMOEA/D algorithm. Then, several experiments can demonstrate the superiority of post-innovation AEA over pre-innovation AMOEA/D. According to the experiments, the effectiveness of my innovation can be verified.

In this paper, LSMOP benchmark problems are adopted to evaluate the performance of AEA. For each test problem in LSMOP, the number of decision variables is set as n = 100, 300, 500, 1000. Moreover, the number of objectives for each problem in LSMOP is set to m = 2. Please note that other parameters in the LSMOP benchmark suites are set the same as suggested in original references. Here, the well-known inverted generational distance (IGD)[31] is adopted as the performance indicator, since IGD can simultaneously evaluate whether the convergence and diversity of the final solution set are good or not. To calculate IGD, a large number of solutions are evenly sampled from the real PF of the target LSMOP. In our experiments, 5000 and 10000 reference points are evenly sampled from the true PF in solving bi-objective test instances, respectively. Let us suppose that $P^{'}$ is a set of reference points uniformly distributed on the real PF and $R$ is a set of non-dominated solutions selected from the final population. Hence, IGD can be calculated as follow:

$$IGD(P^{'}, R) = \frac{\sum_{p \in P'} dis(p, R)}{|P'|} \tag{5}$$

where $dis(p, R)$ denotes the minimum Euclidean distance between the reference point $p$ and solutions in $R$, and $|P^{'}|$ denotes the size of $P^{'}$. A smaller IGD value indicates that the final population with better convergence and diversity is obtained. In addition, in the following experimental studies, the Wilcoxon rank-sum test with a 0.05 significance level is applied to show statistically significant differences on the IGD results. Symbols "+," "−," and "=" indicate that the result of the compared algorithm is, respectively, better than, worse than, and similar to that obtained by AEA in the following tables.

### 4.3 Main contributions

- An adaptive two-stage strategy is proposed in this paper, where the first and second stages focus more on speeding up convergence and enhancing diversity of the evolutionary population, respectively. Once the population is detected to be evolutionarily stagnated in the first stage, the second stage will be activated for remedying the population's diversity. In this way, AEA can fully realize the promising performance complementarity between accelerating evolutionary search and maintaining population diversity when solving LMOPs with different difficulties.
- An accelerating optimizer is designed in the first stage of AEA to speed up the population's convergence. To be specific, an artificial neural network (ANN) is trained by using paired low-quality solutions and high-quality solutions, respectively, acting as the input and the expected output in the prepared training data.
- A CSO is designed in the second stage of AEA, aiming to remedy the population's diversity obtained in the first stage.

To clearly illustrate the proposed algorithm, Algorithm 2 gives the details. In detail, an initial population $P$ with $N$ solutions is randomly generated in the search space of the target LMOP, $N$ is the population size. Besides, the set of weight vectors, represented by $V$, are generated using the uniform method[4] in the objective space, as shown in line 1 of Algorithm 2. Then, the evaluation function ($FE$) counter is initialized to 0, the preset threshold , used to adaptively control the switch from the first stage to the second stage, is set to 0.1, and the symbol flag is set to false in line 2. Particularly, the first stage is executed when flag is false and the second stage is activated when flag is true. Then, the main loop of the evolutionary process is repeatedly run from line 3 to line 11, until the termination condition is satisfied , where $FEmax$ represents the maximum function evaluations preset by the users.

The first evolutionary stage is run from line 4 to line 7. Concretely, the current population $P$ is updated in line 5 by an accelerating optimizer (Algorithm 1). This optimizer will also return the parameter $eNum$, which indicates the number of solutions in $P$ that are dominated by the newly generated offspring in the accelerating optimizer. Then, the current evolutionary status is discriminated based on the value of $eNum$ in line 6. If the evolutionary status is determined to be stagnated (i.e., $eNum/N < 0.15$), the symbol flag is changed to true, which represents the following evolutionary process will switch to the second stage in line 10. The condition $eNum/N < 0.15$ indicates that the population is detected to be evolutionarily stagnated or evolved with a very slow speed. Thus, it will waste computing resources if the evolutionary process is still stayed in the first stage. To avoid this issue, the population $P$ obtained in the first stage is further updated in the second stage by a classical competitive swarm optimizer[27]in line 10.

**Algorithm 2** The framework of the proposed AEA

```
1:  Initialize the weight vectors V and the current population P;
2:  Initialize flag = false, SR = 0.15 and FE = 0;
3:  while termination criterion is not fulfilled do
4:      if flag == false then
5:          (P, eNum) = Accelerating-Optimizer (P);
6:          if eNum/N < SR then
7:              Set flag = true;
8:          end if
9:      else
10:         P = CSO (P, V);
11:     end if
12:     FE = FE + N;
13: end while
14: return P;
```

## 5 Results and analysis

To r validate the performance of AEA, AEA is compared with AMOEA/D in handling LSMOP1-LSMOP9 in the case of $m = 2$ and $n = (100, 300, 500, 1000)$. The experimental results obtained by AEA and AMOEA/D are presented in Table I, in which the best result compared with the others is highlighted in bold. As observed from Table I, AEA obtains 34 out of 36 best results and AMOEA/D only gets 2 out of 36 best results. Therefore, we can reasonably conclude that AEA performs better than AMOEA/D in solving most of these LSMOP

problems, which can verify the effectiveness of our proposed adaptive strategy.

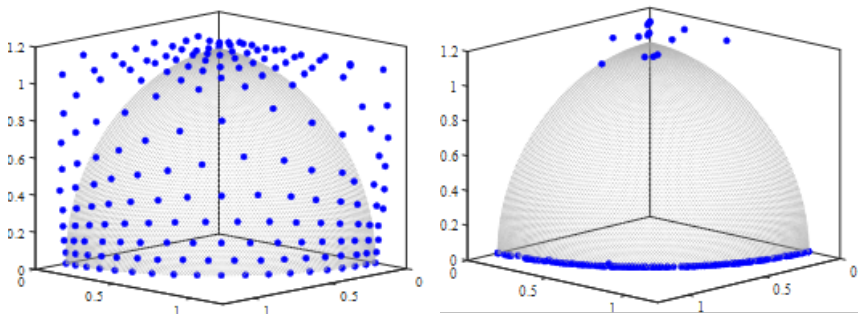
After comparing the performance of AEA and AMOEA/D, the final solutions of the above algorithms on tri-objective LSMOP5 with 300 decision variables are depicted in Fig. 2. As observed from Fig. 2 (a), only AEA can obtain the final population to evenly cover the true PF. According to Fig. 2 (b), For AMOEA/D, it may waste some computational resources, as it cannot adaptively judge the switch of two evolutionary stages. Therefore, it may trap into local optima. In contrast, our proposed algorithm AEA can avoid the above shortcomings by using an adaptive two-stage strategy and designing two improved optimizers for different stages.

Table 1: The IGD results obtained by AMOEA/D and AEA on 2-objective LSMOP benchmarks

| Problem | $n$ | AMOEA/D | AEA |
|---|---|---|---|
| LSMOP1 | 100 | 2.113E-1(2.98E-2)- | **4.097E-3(2.34E-4)** |
| | 300 | 2.118E-1(2.62E-2)- | **5.285E-2(2.07E-2)** |
| | 500 | 2.401E-1(2.53E-2)- | **1.337E-1(6.98E-3)** |
| | 1000 | 3.283E-1(2.76E-2)- | **1.712E-1(7.24E-3)** |
| LSMOP2 | 100 | 5.540E-2(1.22E-2)- | **1.099E-2(1.59E-3)** |
| | 300 | 1.614E-2(6.32E-4)- | **1.422E-2(3.21E-3)** |
| | 500 | 1.176E-2(4.40E-4)- | **1.018E-2(1.02E-3)** |
| | 1000 | 8.792E-3(1.77E-3)= | **7.836E-3(1.48E-3)** |
| LSMOP3 | 100 | 8.118E-1(6.77E-2)- | **5.732E-1(7.41E-2)** |
| | 300 | 1.239E+0(2.21E-1)= | **1.039E+0(1.16E-1)** |
| | 500 | 1.399E+0(1.77E-1)= | **1.275E+0(1.52E-1)** |
| | 1000 | 1.568E+0(1.97E-2)- | **1.454E+0(1.13E-1)** |
| LSMOP4 | 100 | 2.271E-2(1.54E-3)- | **6.972E-3(9.50E-4)** |
| | 300 | 2.660E-2(2.94E-3)- | **8.712E-3(2.27E-4)** |
| | 500 | 2.552E-2(1.63E-3)- | **8.757E-3(1.31E-3)** |
| | 1000 | 1.858E-2(4.58E-4)- | **1.250E-2(8.82E-4)** |
| LSMOP5 | 100 | 3.586E-2(1.13E-2)- | **4.303E-3(6.21E-5)** |
| | 300 | 3.486E-1(5.99E-2)- | **1.455E-2(1.44E-3)** |
| | 500 | 6.457E-1(1.77E-2)- | **8.289E-2(1.44E-2)** |
| | 1000 | 6.614E-1(1.77E-2)- | **4.274E-1(6.64E-2)** |
| LSMOP6 | 100 | 4.095E-1(3.10E-2)- | **3.367E-1(2.25E-2)** |
| | 300 | **4.420E-1(1.20E-2)=** | 4.444E-1(1.11E-2) |
| | 500 | 4.311E-1(1.41E-2)= | **4.089E-1(5.58E-2)** |
| | 1000 | 4.057E-1(1.40E-3)= | **3.866E-1(6.55E-2)** |
| LSMOP7 | 100 | 1.360E+0(1.62E-1)- | **8.554E-1(4.51E-2)** |
| | 300 | 1.498E+0(2.38E-3)= | **1.483E+0(3.41E-2)** |
| | 500 | 1.510E+0(1.09E-3)- | **1.476E+0(9.41E-2)** |
| | 1000 | 1.516E+0(3.02E-3)= | **1.514E+0(1.71E-3)** |
| LSMOP8 | 100 | 5.179E-2(7.02E-3)- | **4.753E-3(5.44E-4)** |
| | 300 | 4.186E-2(4.38E-3)- | **1.897E-2(1.62E-3)** |
| | 500 | 7.673E-2(1.58E-2)- | **1.893E-2(6.65E-4)** |
| | 1000 | 3.857E-1(7.83E-2)- | **3.768E-2(4.39E-3)** |
| LSMOP9 | 100 | **7.445E-1(1.38E-1)=** | 8.107E-1(1.86E-3) |
| | 300 | 7.982E-1(4.18E-2)- | **7.828E-1(8.57E-2)** |
| | 500 | 8.136E-1(1.35E-2)= | **8.089E-1(5.23E-4)** |
| | 1000 | 8.271E-1(2.93E-2)- | **7.233E-1(1.16E-1)** |
| +/-/= | | 0/26/10 | — |

## 6 Conclusion and future work

In this paper, an evolutionary algorithm with an adaptive two-stage search strategy has been proposed for large-scale multi-objective optimization, called AEA. In the proposed optimizer, the two evolutionary stages are adaptively adjusted according to the characteristic of LMOPs. Specifically, the adaptive accelerating optimizer is used in the first stage to accelerate the convergence speed, while the CSO optimizer is run in the second stage to diversify the quasi-optimal solutions obtained in the first stage to cover the PF. Moreover, we do



(a) (b)

Figure 2: The final solutions obtained by AEA and AMOEA/D on tri-objective LSMOP5 with 300 decision variables, With the true PF indicated by the shaded area.

the comparison experiments between the pre-innovation algorithm AMOEA/D and post-innovation algorithm AEA, which can demonstrate the superiority of our innovation.

In our future work, to train a more suitable model, the selection of training mode (i.e., machine learning or deep learning models) into LMOEA and the construction of training model will be further studied for various LMOPs. In addition, it is also interesting to apply the proposed AEA for solving complete and real combinatorial problems to test the proposed method. Finally, in the follow-up research, we focus on how to improve the search speed and performance of AEA when dealing with higher dimensional problems with more than 1000 decision variables.

## References


[1] ISHIBUCHI H, MURATA T. A multi-objective genetic local search algorithm and its application to flowshop scheduling[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 1998, 28(3): 392-403. DOI: 10.1109/5326.704576.

[2] BANDYOPADHYAY S, MUKHERJEE A. An Algorithm for Many-Objective Optimization With Reduced Objective Computations: A Study in Differential Evolution[J]. IEEE Transactions on Evolutionary Computation, 2015, 19(3): 400-413. DOI: 10.1109/TEVC.2014.2332878.

[3] TIAN Y, SI L, ZHANG X, et al. Evolutionary Large-Scale Multi-Objective Optimization: A Survey[J]. J. ACM, 2021, 1(1).

[4] ZHANG Q, LI H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition[J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731. DOI: 10.1109/TEVC.2007.892759.

[5] TIAN Y, CHENG R, ZHANG X, et al. An Indicator-Based Multiobjective Evolutionary Algorithm With Reference Point Adaptation for Better Versatility[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(4): 609-622. DOI: 10.1109/TEVC.2017.2749619.

[6] CHEN H, TIAN Y, PEDRYCZ W, et al. Hyperplane Assisted Evolutionary Algorithm for Many-Objective Optimization Problems[J]. IEEE Transactions on Cybernetics, 2020, 50(7): 3367-3380. DOI: 10.1109/TCYB.2019.2899225.

[7] SHANG K, ISHIBUCHI H. A New Hypervolume-Based Evolutionary Algorithm for Many-Objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2020, 24(5): 839-852. DOI: 10.1109/TEVC.2020.2964705.

[8] LIN Q, LIU S, ZHU Q, et al. Particle Swarm Optimization With a Balanceable Fitness Estimation for Many-Objective Optimization Problems[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(1): 32-46. DOI: 10.1109/TEVC.2016.2631279.

[9] WANG H, JIAO L, SHANG R, et al. A Memetic Optimization Strategy Based on Dimension Reduction in Decision Space[J]. Evolutionary Computation, 2015, 23(1): 69-100. DOI: 10.1162/EVCO_a_00122.

[10] PARSONS L, HAQUE E, LIU H. Subspace clustering for high dimensional data: a review[J]. Acm sigkdd explorations newsletter, 2004, 6(1): 90-105.

[11] CHENG R, JIN Y, OLHOFER M, et al. Test Problems for Large-Scale Multiobjective and Many-Objective Optimization[J]. IEEE Transactions on Cybernetics, 2017, 47(12): 4108-4121. DOI: 10.1109/TCYB.2016.2600577.

[12] MA X, LIU F, QI Y, et al. A Multiobjective Evolutionary Algorithm Based on Decision Variable Analyses for Multiobjective Optimization Problems With Large-Scale Variables[J]. IEEE Transactions on Evolutionary Computation, 2016, 20(2): 275-298. DOI: 10.1109/TEVC.2015.2455812.

[13] ZHANG X, TIAN Y, CHENG R, et al. A Decision Variable Clustering-Based Evolutionary Algorithm for Large-Scale Many-Objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(1): 97-112. DOI: 10.1109/TEVC.2016.2600642.

[14] ZILLE H, ISHIBUCHI H, MOSTAGHIM S, et al. A Framework for Large-Scale Multiobjective Optimization Based on Problem Transformation[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(2): 260-275. DOI: 10.1109/TEVC.2017.2704782.

[15] HE C, LI L, TIAN Y, et al. Accelerating Large-Scale Multiobjective Optimization via Problem Reformulation[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(6): 949-961. DOI: 10.1109/TEVC.2019.2896002.

[16] QIN S, SUN C, JIN Y, et al. Large-Scale Evolutionary Multiobjective Optimization Assisted by Directed Sampling[J]. IEEE Transactions on Evolutionary Computation, 2021, 25(4): 724-738. DOI: 10.1109/TEVC.2021.3063606.

[17] HE C, CHENG R, YAZDANI D. Adaptive Offspring Generation for Evolutionary Large-Scale Multiobjective Optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020: 1-13. DOI: 10.1109/TSMC.2020.3003926.

[18] HONG W, TANG K, ZHOU A, et al. A Scalable Indicator-Based Evolutionary Algorithm for Large-Scale Multiobjective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(3): 525-537. DOI: 10.1109/TEVC.2018.2881153.

[19] CHENG R, JIN Y. A Competitive Swarm Optimizer for Large Scale Optimization[J]. IEEE Transactions on Cybernetics, 2015, 45(2): 191-204. DOI: 10.1109/TCYB.2014.2322602.

[20] RUMELHART D E, HINTON G E. Williams "learning internal representations by error propagation" in" parallel distributed processing[M]. Mechanisms of Sentence Processing: Assigning roles to constituents of sentences, 1986.

[21] ZHANG X, DU K J, ZHAN Z H, et al. Cooperative Coevolutionary Bare-Bones Particle Swarm Optimization With Function Independent Decomposition for Large-Scale Supply Chain Network Design With Uncertainties[J]. IEEE Transactions on Cybernetics, 2020, 50(10): 4454-4468. DOI: 10.1109/TCYB.2019.2937565.

[22] GE H, SUN L, TAN G, et al. Cooperative Hierarchical PSO With Two Stage Variable Interaction Reconstruction for Large Scale Optimization[J]. IEEE Transactions on Cybernetics, 2017, 47(9): 2809-2823. DOI: 10.1109/TCYB.2017.2685944.

[23] YANG Q, CHEN W N, GU T, et al. A Distributed Swarm Optimizer With Adaptive Communication for Large-Scale Optimization[J]. IEEE Transactions on Cybernetics, 2020, 50(7): 3393-3408. DOI: 10.1109/TCYB.2019.2904543.

[24] LAN R, ZHU Y, LU H, et al. A Two-Phase Learning-Based Swarm Optimizer for Large-Scale Optimization[J]. IEEE Transactions on Cybernetics, 2020: 1-10. DOI: 10.1109/TCYB.2020.2968400.

[25] ZHANG W X, CHEN W N, ZHANG J. A dynamic competitive swarm optimizer based-on entropy for large scale optimization[C]//2016 Eighth International Conference on Advanced Computational Intelligence (ICACI). 2016: 365-371. DOI: 10.1109/ICACI.2016.7449853.

[26] YANG Q, CHEN W N, DENG J D, et al. A Level-Based Learning Swarm Optimizer for Large-Scale Optimization[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(4): 578-594. DOI: 10.1109/TEVC.2017.2743016.

[27] TIAN Y, ZHENG X, ZHANG X, et al. Efficient Large-Scale Multiobjective Optimization Based on a Competitive Swarm Optimizer[J]. IEEE Transactions on Cybernetics, 2020, 50(8): 3696-3708. DOI: 10.1109/TCYB.2019.2906383.

[28] WANG X, ZHANG K, WANG J, et al. An Enhanced Competitive Swarm Optimizer with Strongly Convex Sparse Operator for Large-Scale Multi-Objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2021.

[29] LIU S, LIN Q, LI Q, et al. A Comprehensive Competitive Swarm Optimizer for Large-Scale Multiobjective Optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021: 1-14. DOI: 10.1109/TSMC.2021.3131312.

[30] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197. DOI: 10.1109/4235.996017.

[31] ZHOU A, JIN Y, ZHANG Q, et al. Combining Model-based and Genetics-based Offspring Generation for Multi-objective Optimization Using a Convergence Criterion[C]/ /2006 IEEE International Conference on Evolutionary Computation. 2006: 892-899. DOI: 10.1109/CEC.2006.1688406.