Memetic Computing
https://doi.org/10.1007/s12293-022-00368-7

REGULAR RESEARCH PAPER

Adaptive multiobjective evolutionary algorithm for large-scale transformer ratio error estimation

Changwu Huang[1] · Lianghao Li[2] · Cheng He[1] · Ran Cheng[1] · Xin Yao[1]

Received: 7 June 2021 / Accepted: 4 April 2022

Abstract
As a typical large-scale multiobjective optimization problem extracted from real-world applications, the voltage transformer ratio error estimation (TREE) problem is challenging for existing evolutionary algorithms (EAs). Due to the large number of decision variables in the problems, existing algorithms cannot solve TREE problems efficiently. Besides, most EAs may fail to balance the convergence enhancement and diversity maintenance, leading to the trap in local optima even at the early stage of the evolution. This work proposes an adaptive large-scale multiobjective EA (LSMOEA) to handle the TREE problems with thousands of decision variables. Generally, multiple efficient offspring generation and environmental selection strategies selected from some representative LSMOEAs are included. Then an adaptive selection strategy is used to determine which offspring generation and environmental selection operators are used in each generation of the evolution. Thus, the search behavior of the proposed algorithm evolves along with the evolution process, the balance between convergence and diversity is maintained, and the proposed algorithm is expected to solve TREE problems effectively and efficiently. Experimental results show that the proposed algorithm achieves significant performance improvement due to the adaptive selection of different operators, providing an effective and efficient approach for large-scale optimization problems.

Keywords Large-scale optimization · Multiobjective optimization · Adaptive operator selection · Voltage transformer ratio error estimation

1 Introduction

Multiobjective optimization problems (MOPs) involve multiple, often conflicting, objectives that need to be optimized simultaneously [7]. There are many MOPs in real-world applications, such as flowshop scheduling [26], portfolio optimization [36], and voltage transformers ratio error estimation (TREE) [21]. Unlike single-objective optimization problems, there is a set of trade-off optimal solutions instead of a single optimum in an MOP. Specifically, these trade-off optima constitute the Pareto optimal set (PS), and the projection of the PS in the objective space forms the Pareto optimal front (PF) [35]. Due to their population-based characteristics for obtaining multiple solutions in a single run, evolutionary algorithms (EAs) are naturally efficient and suitable for

This work was supported by the National Natural Science Foundation of China (Nos. U20A20306, 61903178, and 61906081), the Guangdong Basic and Applied Basic Research Foundation (No. 2019A1515110575), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), and the Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531 and RCBS20200714114817264).

✉ Cheng He
chenghehust@gmail.com

Changwu Huang
huangcw3@sustech.edu.cn

Lianghao Li
lianghaoli.hust@gmail.com

Ran Cheng
ranchengcn@gmail.com

Xin Yao
xiny@sustech.edu.cn

1 Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China

2 Key Laboratory of Image Information Processing and Intelligent Control of Education Ministry of China, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China

Published online: 04 May 2022

multiobjective optimization [24]. In recent decades, a variety of multiobjective EAs (MOEAs) have been proposed, including the Pareto-based MOEAs (e.g., NSGA-II [12] and SPEA2 [51]), the indicator-based MOEAs (e.g., IBEA [49] and SMS-EMOA [3]), and the decomposition-based MOEAs (e.g., MOEA/D [45] and MOEA/D-M2M [31]). Existing MOEAs have shown to be effective in solving conventional MOPs, but their performance drops drastically when the number of decision variables increases to hundreds or even more.

MOPs with a large number of decision variables, usually more than 100, are also known as large-scale MOPs (LSMOPs) [5,6]. With the linear increment in the number of decision variables, the search space's volume and complexity will increase exponentially. Using a population of limited size to search an enormous search space with an acceptable computational cost is the main challenge of solving LSMOPs. Recently, with a growing interest in large-scale multiobjective optimization, four main types of approaches are proposed to tackle LSMOPs [42]. The cooperative coevolution (CC) framework based approaches formed the first type, which handles large-scale decision variables in a divide-and-conquer manner [37]. Specifically, the CC method decomposes the decision variables into several groups and cooperatively optimizes all the groups of decision variables, e.g., third-generation CC differential evolution algorithm [1]. The second type of approach reformulates the original LSMOPs into simpler problems with low-dimensional decision variables. For instance, the weighted optimization framework used decision variable grouping methods and some transforming functions to reduce the number of decision variables [47]; the large-scale multiobjective optimization framework (LSMOF) [23] uses problem reformulation to improve the efficiency of MOEAs on LSMOPs. The third type of approach is dedicated to the decision variable analysis based ones, e.g., MOEA based on decision variable analysis (MOEA/DVA) [32] and decision variable clustering-based large-scale EA (LMEA) [46]. The last type of approach tries to enhance the effectiveness of offspring generation in conventional MOEAs, i.e., generating evenly distributed offspring solutions with better convergence than conventional MOEAs. Typical algorithms include the competitive swarm optimizer based EA (LMOCSO) [43], the direction guided adaptive offspring generation based EA (DGEA) [20], and the directed sampli ng assisted EA (LMOEA-DS [38]).

Existing LSMOEAs have shown their advantages in terms of either efficiency or effectiveness by using different decision variable handling strategies, leading to their success in solving some benchmark problems (e.g., ZDT problems [48], DTLZ problems [13], WFG problems [25], and LSMOP problems [6]). However, their performance could be unsatisfactory in solving real-world applications, such as TREE [21]. It could be time-consuming and error-prone to design a specific LSMOEA for solving TREE problems. A promising and practical way is to use existing LSMOEAs to design new algorithms. We can use the evolutionary operators (i.e., offspring generation operator and environmental selection operator) in existing powerful MOEAs to design new algorithms for the problems at hand. Many attempts have been carried out under this line, and several mixed or hybrid MOEAs which combine different evolutionary operators have been proposed to solve MOPs and other optimization problems [4,30,44]. One essential and challenging task in designing a hybrid or mixed EA (i.e., an EA constructed by operators from different EAs) is to select suitable evolutionary operators to be used in the mixed EA. This task is usually referred to as an operator selection problem in the evolutionary computation community. The operator selection problem aims at identifying the most proper operators from several candidate operators to be used, and thus the algorithm can perform well on the given problems [9]. Adaptive operator selection (AOS) is a popular and efficient category of approaches to address operator selection problem [29]. AOS approaches solve operator selection problems in an online fashion, i.e., it dynamically selects suitable operators during the problem-solving process or the run of the algorithm, rather than identifying proper operators before adopting the algorithm to solve the problem. Due to the adaptive and dynamic properties of AOS approaches, they are suitable for handling complex LSMOPs. Motivated by this, we design an adaptive LSMOEA using adaptive operator selection, abbreviated to AOS-LSMOEA hereafter, to solve the TREE problems. Generally, the proposed AOS-LSMOEA approach can use the crucial components or operators in existing LSMOEAs and adaptively select suitable ones from them to use. It can enhance the effectiveness of existing LSMOEAs in solving TREE problems and be applicable to other black-box LSMOPs.

In the remainder of this work, we first introduce some background about the AOS-LSMOEA, including the TREE problems, the iterated problem reformulation based large-scale MOEA (iLSMOA), and the AOS, in Sect. 2. The schema and details of our proposed AOS-LSMOEA approach are given in Sect. 3, and the experimental studies are provided in Sect. 4. Finally, conclusions are drawn in Sect. 5.

2 Background

In this part, we first briefly introduce the formulation of the involved TREE problems. Next, the offspring generation and environmental selection operators for large-scale multiobjective optimization are elaborated. Afterwards, we illustrate the background of adaptive operator selection. Finally, a short introduction to the iterated problem reformulation based large-scale MOEA (iLSMOA) [19] is given. Notably, iLSMOA is the basis of the proposed AOS-LSMOEA.

2.1 TREE problems

Unlike existing artificial benchmark problems in the field of large-scale multiobjective optimization, TREE problems are extracted from real-world applications with data sampled from different substations in the power delivery system [21]. Generally, TREE problems transform conventional voltage transformer (VT) ratio error estimation tasks into large-scale multiobjective optimization. (1) Physical and statistical rules are adapted to construct objectives related to the measured voltage values. (2) The decision variables in TREE problems are the true voltage values of the power delivery system over time. Due to irregular data sampled from the power delivery system, the PS and PF of the TREE problems are irregular, leading to challenges in diversity maintenance in both the decision and objective spaces for MOEAs. The detailed formulations of two types of objectives are given.[1]

Let us denote the sampled data from the ith VT is $\boldsymbol{d}^i = (d^i_{1,1}, \ldots, d^i_{1,T}, \ldots, d^i_{K,1}, \ldots, d^i_{K,T})$, where $d^i_{p,q}$ denotes the measured data from pth phase of the ith VT at time q. Usually, there are three primary/secondary or six phases (including three primary and three secondary phases), i.e., K is three or six. Meanwhile, the ground truth voltage values can be denoted as $\boldsymbol{x} = (x_{1,1}, \ldots, x_{1,T}, \ldots, x_{K,1}, \ldots, x_{K,T})$. Then the time-varying ratio error of the ith VT $\boldsymbol{e}^i$ is $(\frac{d^i_{1,1}-x_{1,1}}{x_{1,1}}, \ldots, \frac{d^i_{1,T}-x_{1,T}}{x_{1,T}}, \ldots, \frac{d^i_{K,1}-x_{K,1}}{x_{K,1}}, \ldots, \frac{d^i_{K,T}-x_{K,T}}{x_{K,T}})$. Without loss of generality, it can be simplified as $\boldsymbol{e}^i = (e^i_{1,1}, \ldots, e^i_{1,T}, \ldots, e^i_{K,1}, \ldots, e^i_{K,T})$. In this case, we can form the first objective function, where the goal is to minimize the total time-varying ratio errors of all the VTs. We can have

$$f_1(\boldsymbol{x}, \boldsymbol{e}^1, \ldots, \boldsymbol{e}^P) = \sum_{j=1}^{T}\sum_{k=1}^{K}\sum_{i=1}^{P} e^i_{k,j}. \tag{1}$$

Figure 1 shows the irregular data sampled from a substation with 12 sets of VTs, where each VT set includes three primary voltage phases. The horizontal axis is the length of the measured sequential data, and the vertical axis shows the measured primary voltage value.

Accordingly, it can be derived from (1) that this function is fully separable, i.e., no decision variable is interacting with

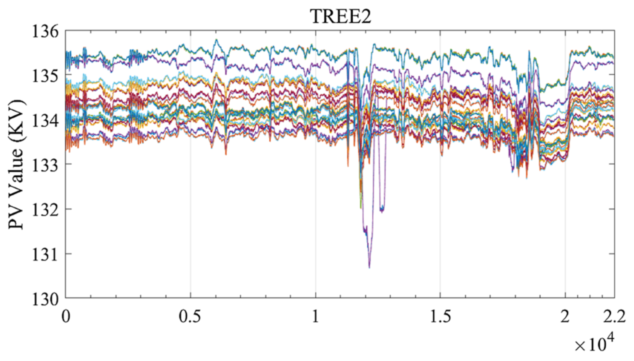


Fig. 1 The measured primary voltage values in a substation with 12 sets of VTs (each set contains three VTs, and thus a total number of 36 data sequences are involved

any others [34]. Consequently, it leaves space for decision variable analysis or CC-based methods to efficiently handle the large-scale decision variables.

Once the variation of errors over time is considered, we can obtain the second objective function

$$f_2(\boldsymbol{x}, \boldsymbol{\Delta e}^1, \ldots, \boldsymbol{\Delta e}^P) = \sum_{i=1}^{P}\sqrt{std((\Delta e^i_{1,1}, \ldots, \Delta e^i_{K,T}))}, \tag{2}$$

where $\boldsymbol{\Delta e}^i$ is $(e^i_{1,2} - e^i_{1,1}, \ldots, e^i_{1,T} - e^i_{1,T-1}, \ldots, e^i_{K,2} - e^i_{K,1}, \ldots, e^i_{K,T} - e^i_{K,T-1})$ with $std(*)$ denoting the standard deviation of vector $*$. Unlike objective f_1, the construction of f_2 is more complicated, involving different variable interaction relationships. For TREE1 to TREE5, there is no independent decision variable, and all the decision variables can be grouped into three groups of equal size. As a result, improper grouping results may lead to a degenerated performance in CC-based methods. Regarding TREE6 and TREE7, there are both independent and interacting decision variables, which is challenging for CC-based and decision variable analysis based algorithms.

The relationship between each decision variable and all the objectives is also nontrivial. Suppose that each decision variable can be classified as a position or a distance variable as defined in [32,46]. All the decision variables are position variables in TREE1 to TREE5, and only half of the decision variables are position variables in TREE6 and TREE7. TREE problems involve a large number of decision variables, irregular PFs and PSs, and complex variable interactions, showing the difficulty of real-world LSMOPs and pointing out some open problems for the design of large-scale MOEAs.

[1] In the original TREE problems, several constraints and the third objective are also designed. Since the topic of this work is unconstrained large-scale multiobjective optimization, the constraints and the third objective are ignored in this work. Moreover, we have constructed the TREE7 problem from a substation with the same topology as TREE6. With different decision variables and sampled data, TREE7 is the same as TREE6 in function formulation.

2.2 Environmental selection and offspring generation

Offspring generation and environmental selection are two main components of EAs. Several MOEAs with different environmental selection approaches have been proposed in recent decades. They can be broadly classified into three categories, i.e., dominance-based MOEAs, decomposition-based MOEAs, and indicator-based MOEAs.

Dominance-based MOEAs compare candidate solutions according to Pareto dominance (e.g., NSGA-II [12] and PESAII [10]) or some modified dominance relationship (e.g., SPEA2 [51]). Assisted by the reference/weight points/weights, the decomposition-based MOEAs decompose an MOP into a series of subproblems and optimize them simultaneously (e.g., MOEA/D [45] and NSGA-III [11]). In the indicator-based MOEAs, some indicators are adopted to assess the contribution of solutions to the whole population, and the candidate solutions are selected accordingly (e.g., IBEA [49]). Generally, existing MOEAs are effective in environmental selection since only two to three objectives are involved.

In addition to the environmental selection, the offspring generation also plays a vital role in evolutionary multi-objective optimization [22]. Generally, two representative offspring generation strategies in conventional MOEAs and three in large-scale MOEAs are involved in this work. The first two offspring generation strategies are the simulated binary crossover and polynomial mutation-based one in NSGA-II [12] and the differential evolution based one in MOEA/D-DE [28]. The rest three strategies are the problem reformulation based one in LSMOF [23], the direction guided one in DGEA [20], and the competitive swarm optimization based one in LMOCSO [43]. Notably, the third evolution step of generalized differential evolution (GDE3) [27] also used the differential evolution operator, and a particle swarm optimization based strategy is used in SMPSO [33]. Due to the similarity and efficiency considerations of offspring generation strategies, we will not introduce others.

2.3 Adaptive operator selection

AOS is a recent paradigm to tackle the operator selection problem of EAs in an adaptive manner or online fashion. AOS aims to select a suitable operator from the given candidate operators set to use during the search process. Usually, the selection is conducted based on the recent performance or some measure of the quality of candidate operators. AOS mainly involves two components: (1) credit assignment, which defines how to assign credit or reward to an operator based on the impact brought by its recent application on the current search process; and (2) operator selection, which selects the operator to be applied next based on the previously collected rewards. These two components used in this work are described in the following.

2.3.1 Credit assignment

The goal of credit assignment is to provide a reward to an operator after it has been applied, which is based on its performance regarding the progress of the search. Most credit assignment methods use fitness improvement as the reward [15,16]. Since we apply AOS to an LSMOEA, the fitness improvement cannot be used directly as a reward in credit assignments. We propose to adopt the hypervolume (HV) indicator [50], which is widely used to measure the performance of MOEAs, for measuring the impact of operator application, and the improvement of HV is taken as the reward in credit assignment.

At the beginning of generation or iteration g, the HV value of the population is calculated with respect to the reference point and denoted as $\mathrm{HV}^{(g)}$. Here, the ideal point of the initial population forms the reference point for calculating the HV value. Then, the evolutionary operations are performed to evolve the population, and the HV value of the updated population is calculated as $\mathrm{HV}^{(g+1)}$. Afterwards, the reward of evolutionary operator application at generation g is measured by the HV improvement $R^{(g)} = \max\left((\mathrm{HV}^{(g+1)} - \mathrm{HV}^{(g)}), 0\right)$, and then $R^{(g)}$ is taken as the reward for the operator used in generation g.

2.3.2 Operator selection

Based on the rewards or credit values received from the credit assignment, the operator selection scheme maintains an up-to-date empirical quality estimate for each candidate operator to update candidate operators' application rates. The most promising and commonly used operator selection schemes are the probability matching (PM) [17], the adaptive pursuit (AP) [39], and the upper confidence bound (UCB) multi-armed bandit algorithm [2,18]. We intend to adopt the AP method for operator selection due to its superior performance over the PM [40], and the PM and UCB algorithms will be tested and compared with AP. In what follows, these three algorithms are briefly described.

The PM scheme can be mathematically formalized as follows. Suppose we have a set of K candidate operators $\mathcal{A} = \{a_1, \ldots, a_K\}$, PM maintains a probability vector $\mathbf{P}^{(g)} = [P_1^{(g)}, \ldots, P_K^{(g)}]$ $(0 \le P_i^{(g)} \le 1$ and $\sum_{i=1}^{K} P_i^{(g)} = 1)$ and a quality vector $\mathbf{Q}^{(g)} = [Q_1^{(g)}, \ldots, Q_K^{(g)}]$ at time g (in this work time is equivalent to iteration or generation count of evolutionary process), where $P_i^{(g)}$ and $Q_i^{(g)}$ are the selection probability and empirical quality estimate of the candidate operator $a_i \in \mathcal{A}$, respectively. Initially, the probability and quality vectors are initialized as $P_i^{(0)} = \frac{1}{K}$, $Q_i^{(0)} = 1.0$

for $i = 1, \ldots, K$. At generation g, the j-th operator a_j is selected with probability $P_j^{(g)}$ through a proportional Selection scheme, such as roulette-wheel selection scheme, i.e.,

$$a_j \leftarrow \text{RouletteWheelSelection}(\mathbf{P}^{(g)}). \tag{3}$$

After the application of operator a_j, a reward $R_j^{(g)}$ is obtained by the credit assignment. Then $Q_j^{(g)}$ is updated to account for the currently received reward $R_j^{(g)}$, via the exponential recency-weight average updating mechanism, as,

$$Q_j^{(g+1)} = Q_j^{(g)} + \alpha \left[R_j^{(g)} - Q_j^{(g)} \right], \tag{4}$$

where α $(0 < \alpha \leq 1)$ is the learning rate. Then, the selection probability of each candidate operators are calculated as

$$P_i^{(g+1)} = P_{min} + (1 - K \cdot P_{min}) \frac{Q_i^{(g+1)}}{\sum_{j=1}^{K} Q_j^{(g+1)}} \quad \text{for } i = 1, \ldots, K, \tag{5}$$

where $P_{\min}$ is the minimal selection probability value which avoids the selection probability decreases to zero. The complete procedure of PM is presented in Algorithm 1.

Algorithm 1 Probability Matching (K, P_{min}, α)

```
1: for $i = 1$ to $K$ do
2:     $P_i \leftarrow \frac{1}{K}$; $Q_i \leftarrow 1.0$.
3: end for
4: while NotTerminated do
5:     if one or more operators not applied yet then
6:         $a_j \leftarrow$ uniformly selected between the operators not applied.
7:     else
8:         $a_j \leftarrow$ RouletteWheelSelect($\mathbf{P}$).
9:     end if
10:    $R_j \leftarrow$ CreditAssignment.GetReward($a_j$).
11:    $Q_j \leftarrow Q_j + \alpha[R_j - Q_j]$.
12:    for $i = 1$ to $K$ do
13:        $P_i \leftarrow P_{min} + (1 - K \cdot P_{min}) \frac{Q_i}{\sum_{j=1}^{K} Q_j}$.
14:    end for
15: end while
```

The procedures of AP and PM algorithms are similar, but their selection probability update methods are different. In AP, instead of updating the probability vector $\mathbf{P}$ proportionally to the estimated quality vector $\mathbf{Q}$, a winner-takes-all strategy is adopted to increase the selection probability of the current best operator a_{i^*} (where $i^* = \arg\max_i(Q_i^{(g+1)})$) while decreasing the probabilities of the others, as follows:

$$P_{i^*}^{(g+1)} = P_{i^*}^{(g)} + \beta \left[P_{\max} - P_{i^*}^{(g)} \right] \\ P_i^{(g+1)} = P_i^{(g)} + \beta \left[P_{\min} - P_i^{(g)} \right] \quad \text{for } i = 1, \ldots, K \text{ and } i \neq i^*, \tag{6, 7}$$

where $\beta \in [0, 1]$ is the learning rate controlling the greediness of the winner-takes-all strategy, $P_{\min}$ is the minimal selection probability value for avoiding the selection probability decreases to zero, and $P_{\max} = 1 - (K-1)P_{\min}$ is the maximal selection probability. The complete procedure of AP is presented in Algorithm 2.

Algorithm 2 Adaptive Pursuit $(K, P_{min}, \alpha, \beta)$

```
1: $P_{max} \leftarrow 1 - (K-1)P_{min}$.
2: for $i = 1$ to $K$ do
3:     $P_i \leftarrow \frac{1}{K}$; $Q_i \leftarrow 1.0$.
4: end for
5: while NotTerminated do
6:     if one or more operators not applied yet then
7:         $a_j \leftarrow$ uniformly selected between the operators not applied.
8:     else
9:         $a_j \leftarrow$ RouletteWheelSelect($\mathbf{P}$).
10:    end if
11:    $R_j \leftarrow$ CreditAssignment.GetReward($a_j$).
12:    $Q_j \leftarrow Q_j + \alpha[R_j - Q_j]$.
13:    $i^* \leftarrow \arg\max_i(Q_i)$.
14:    for $i = 1$ to $K$ do
15:        if $i = i^*$ then
16:            $P_{i^*} \leftarrow P_{i^*} + \beta[P_{max} - P_{i^*}]$.
17:        else
18:            $P_i \leftarrow P_i + \beta[P_{min} - P_i]$.
19:        end if
20:    end for
21: end while
```

In UCB-based operator selection procedure, at time g each operator a_i has an empirical quality estimate $Q_i^{(g)}$ and a confidence interval. Specifically, the confidence interval depends on the number of times $n_i^{(g)}$ that the operator has been applied before. Then the operator to be used at time $g+1$ is selected by

$$a_j \leftarrow \arg\max_i \left(Q_i^{(g)} + C \sqrt{\frac{2 \ln \left(\sum_{j=1}^{K} n_j^{(g)} \right)}{n_i^{(g)}}} \right), \tag{8}$$

where C is a scaling factor for regulating the trade-off between exploitation and exploration. The complete procedure of UCB is presented in Algorithm 3.

2.4 iLSMOA

In this study, we adopt iLSMOA as the basis to construct the hybridized framework for large-scale multiobjective optimization mainly for two reasons. First, iLSMOA is a flexible framework that can easily involve different components in most existing LSMOEAs. Second, the iterated mechanism in

Algorithm 3 UCB (K, C)

```
1: for $i = 1$ to $K$ do
2:     $n_i \leftarrow 0$; $Q_i \leftarrow 1.0$.
3: end for
4: while NotTerminated do
5:     if one or more operators not applied yet then
6:         $a_j \leftarrow$ uniformly selected between the operators not applied.
7:     else
8:         $a_j \leftarrow \arg\max_i \left( Q_i + C\sqrt{\frac{2\ln\left(\sum_{j=1}^{K} n_j\right)}{n_i}} \right)$.
9:     end if
10:    $R_j \leftarrow$ CreditAssignment.GetReward($a_j$).
11:    $n_j \leftarrow n_j + 1$.
12:    $Q_j \leftarrow \frac{(n_j - 1)Q_j + R_j}{n_j}$.
13: end while
```

iLSMOA is validated to be more effective than its counterpart that uses a two-step strategy in large-scale multiobjective optimization [19].

Algorithm 4 The framework iLSMOA [19].

```
Input: $Z$ (original LSMOP), $N$ (population size), $r$ (number of reference solutions), $g_{\max}$ (maximum iteration).
Output: $\mathcal{P}$ (final population).
1: $\mathcal{P} \leftarrow$ Initialization($N$, $Z$)
2: while NotTerminated do
3:     $Z' \leftarrow$ ProblemReformulation($\mathcal{P}$, $r$, $Z$).
4:     $A \leftarrow$ SingleObjectiveOptimization($Z'$, $g_{\max}$).
5:     $\mathcal{P} \leftarrow$ EnvironmentalSelection($A$, $N$).
6:     $\mathcal{P} \leftarrow$ EvolveByMOEA($\mathcal{P}$, $N$, $g_{\max}$).
7: end while
```

The general framework of iLSMOA is given in Algorithm 4, which uses the problem reformulation based single-objective optimization and conventional MOEA iteratively. To begin with, a population of size N is randomly generated from the original LSMOP Z, and Z is reformulated into a low-dimension single-objective optimization problem Z'. Then the single-objective optimization followed by environmental selection is conducted to obtain a set of well-converged solutions $\mathcal{P}$. Afterwards, $\mathcal{P}$ is further evolved using conventional MOEA for diversity maintenance. Finally, the above procedures are repeated iteratively until the termination criterion is fulfilled. To be more specific, the original iLSMOA uses the MOEA/D-DE [28] during the evolution by MOEA to enhance the diversity of the population.

3 The proposed AOS-LSMOEA

This study proposes the AOS-LSMOEA to solve real-world LSMOPs, i.e., the TREE problems. Multiple specific offspring generation and environment selection operators are included in AOS-LSMOEA to enhance its search ability. Besides, AOS is used to automatically select suitable operators in each iteration (or generation) of the search process. The overall implementation of AOS-LSMOEA is presented in the form of pseudo-code in Algorithm 5.

Algorithm 5 The framework of AOS-LSMOEA.

```
Input: $N$ (population size), $g_{\max}$ (maximum iteration), $\mathcal{P}_{\text{OffGen}}$ (pool of candidate offspring generation opeartors), $\mathcal{P}_{\text{EnvSel}}$ (pool of candidate environmental selection opeators), $P_{\min}$ (minimal selection probability), $\alpha$ and $\beta$ (learning rates).
Output: $\mathcal{P}$ (final population).
1: OffGenSelector $\leftarrow$ AdaptivePursuit($\mathcal{P}_{\text{OffGen}}$, $P_{\min}$, $\alpha$, $\beta$).
2: EnvSelSelector $\leftarrow$ AdaptivePursuit($\mathcal{P}_{\text{EnvSel}}$, $P_{\min}$, $\alpha$, $\beta$).
3: $\mathcal{P} \leftarrow$ Initialization($N$).
4: while NotTerminated do
5:     OffGenOperator $\leftarrow$ Select an operator by OffGenSelector according to (3).
6:     EnvSelOperator $\leftarrow$ Select an operator by EnvSelSelector according to (3).
7:     $A \leftarrow$ OffGenOperator($\mathcal{P}$)
8:     $\mathcal{P} \leftarrow$ EnvSelOperator($A$, $N$).
9:     $\mathcal{P} \leftarrow$ EvolveByMOEA($\mathcal{P}$, $N$, $g_{\max}$).
10:    $R \leftarrow$ Calculate the reward according to credit assignment.
11:    Update quality vector $\mathbf{Q}$ of OffGenSelector and EnvSelSelector according to (4).
12:    Update selection probability vector $\mathbf{P}$ of OffGenSelector and EnvSelSelector according to (6) and (7).
13: end while
```

Given a pool of candidate offspring generation operators $\mathcal{P}_{\text{OffGen}}$ and a pool of candidate environmental selection operators $\mathcal{P}_{\text{EnvSel}}$, two operator selectors (i.e., OffGenSelector and EnvSelSelector) using AP are instantiated respectively for selecting the offspring generation and environmental selection operators. After initializing the population $\mathcal{P}$ of size N, the algorithm enters the main loop. In each generation or iteration, the two selectors first select an offspring generation operator and an environmental selection operator for generating new offspring for updating the population. Next, the reward for the selected operators is calculated according to the credit assignment scheme described in 2.3.1. Then the quality vector of the two selectors is updated based on the received reward (i.e., the improvement of HV) according to the quality updating rules of the AP method described in (4). The probability vector of the two selectors is also updated following (6) and (7). The above-described process is iterated until the termination criterion is fulfilled. The whole process of AOS-LSMOEA is illustrated in Fig. 2.

In this work, the offspring generation strategies used in LSMOF [23], LMOCSO [43], NSGA-II [12], MOEA/D-DE [28], and direction guided offspring generation in DGEA [20] are included as offspring generation operators used in AOS-LSMOEA, i.e., $\mathcal{P}_{\text{OffGen}} = \{$LSMOF, LMOCSO, NSGA-II, MOEA/D-DE, DGEA$\}$. The environmental selection strategies in NSGA-II [12], IBEA [49], and SPEA2 [51] are taken as the possible environment selection

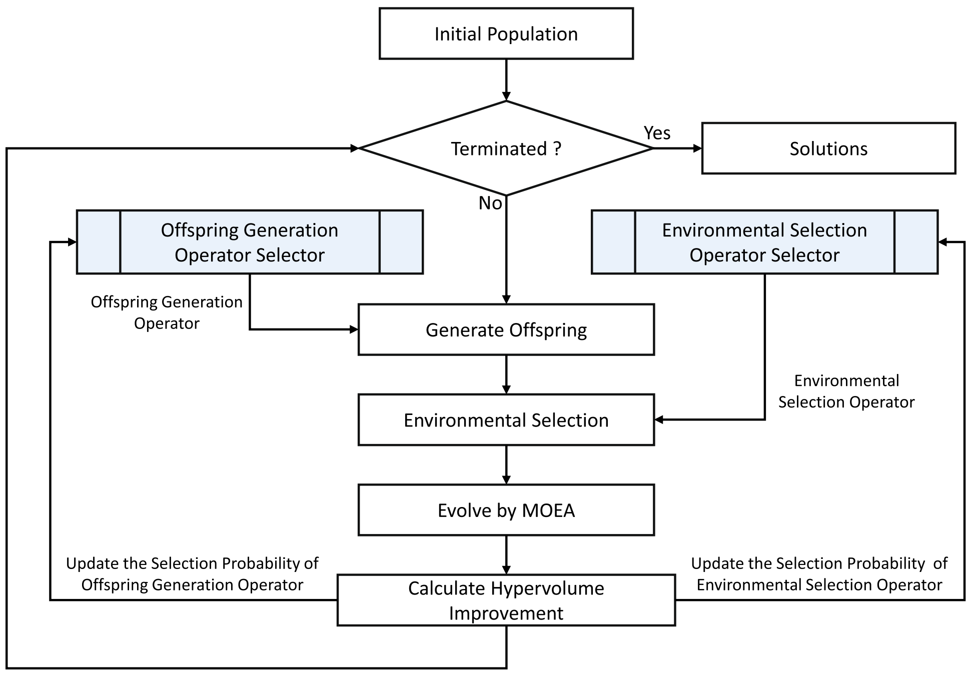

Initial Population
Terminated ?
Yes
Solutions
No
Offspring Generation Operator Selector
Environmental Selection Operator Selector
Offspring Generation Operator
Generate Offspring
Environmental Selection Operator
Environmental Selection
Evolve by MOEA
Update the Selection Probability of Offspring Generation Operator
Calculate Hypervolume Improvement
Update the Selection Probability of Environmental Selection Operator

Fig. 2 Illustration of AOS-LSMOEA using AP

operators as well as without using any selection (WithoutSel), i.e., $\mathcal{P}_{\text{EnvSel}} = \{\text{WithoutSel, NSGA-II, IBEA, SPEA2}\}$. It is worth noting that LSMOF, LMOCSO, and DGEA are representative LSMOEAs, while NSGA-II and MOEA/D-DE are popular MOEAs. The hybridization of these five offspring generation strategies is expected to generate offspring solutions with both good convergence and diversity. Moreover, the environmental selection strategies in NSGA-II, IBEA, and SPEA2 are taken as the possible environment selection strategies and operations without using any selection, aiming to maintain diverse candidate solutions.

4 Experimental studies

To validate the performance of the proposed AOS-LSMOEA, experimental studies on a set of seven TREE problems are performed. Since this study mainly focuses on unconstrained multiobjective optimization, the constraint functions in those TREE problems are ignored. In this section, the settings of experiments are first described, and then the experimental results are presented and discussed.

4.1 Experimental setting

To examine the performance of the proposed AOS-LSMOEA, we conducted a series of comparative experiments. First, the LSMOEA using different operator selection approaches are compared, including AOS-LSMOEA using PM, AP, UCB, uniformly random operator selection strategy, and two mixed strategies. The abbreviations of the investigated AOS-LSMOEAs with operator selection are listed in Table 1.

Second, several powerful and popular MOEAs, including iLSMOA, LMOCSO, DGEA, NSAG-II, and MOEA/D-DE, are compared with the proposed AOS-LSMOEA. Furthermore, AOS-LSMOEA is compared with fifteen hybrid algorithms using different combinations of offspring generation and environmental selection operators. The acronym of the used offspring generation and environmental selection operators are listed in Table 2. The abbreviated name of a hybrid algorithm is denoted as HA(OGi,ESj), which combines offspring generation operator OGi and environmental selection operator ESj.

Each algorithm is run 25 times independently on each test problems. The population size is set as $N = 100$ and the maximal number of fitness function evaluation is set as $MaxFES = 10 \times D$ in all the runs, where D is the dimension of the problem. The parameter values recommended in [39] are used in AP method, i.e., $P_{\min} = 0.05$, $\alpha = 0.8$, and $\beta = 0.8$. For PM, the two parameters are set the same as that in AP, i.e, $P_{\min} = 0.05$ and $\alpha = 0.8$. The scaling factor of UCB is set as $C = 1.0$.

Since we do not know the PFs of real-world test problems, the inverted generational distance (IGD) [8] metric cannot be computed. Thus we use the hypervolume (HV) [50] indicator to evaluate the algorithm's performance. The ideal point of the initial population is used as the reference point, which is used in HV calculation. In comparisons, for each algorithm, the average HV (HV$_{avg}$) and standard deviation of HV (HV$_{std}$) over 25 independent runs on each test problem are presented. Additionally, the box plot of the HV from each

Table 1 The list of compared AOS-LSMOEAs with different operator selection strategies

Acronym	Description
AOS-LSMOEA(AP)	AOS-LSMOEA using AP
AOS-LSMOEA(PM)	AOS-LSMOEA using PM
AOS-LSMOEA(UCB)	AOS-LSMOEA using UCB
UR-LSMOEA	LSMOEA using uniform randomly (UR) operator selection
AOS-LSMOEA(UR,AP)	AOS-LSMOEA where offspring generation operator is uniform randomly selected, and environmental selection operator is selected by AP
AOS-LSMOEA(AP,UR)	AOS-LSMOEA where offspring generation operator is selected by AP, and environmental selection operator is uniform randomly selected

Table 2 The list of used offspring generation and environmental selection operators

Acronym	Offspring generation operator
OG1	Offspring generation strategy used in LSMOF
OG2	Offspring generation strategy used in LMOCSO
OG3	Offspring generation strategy used in NSGA-II
OG4	Offspring generation strategy used in MOEA/D-DE
OG5	Offspring generation strategy used in DGEA
ES0	Without using any environmental selection
ES1	Environmental selection strategy used in NSGA-II
ES2	Environmental selection strategy used in IBEA
ES3	Environmental selection strategy used in SPEA2

Table 3 To investigate the contribution of AOS on offspring generation and environmental selection operators, as well as to further demonstrate the superiority of AOS over uniform random selection strategy, we compared AP with two mixed selection strategies. The results achieved by AOS-LSMOEA(AP), AOS-LSMOEA(UR,AP), and AOS-LSMOEA(AP,UR) Comparison of AOS-LSMOEAs using three different AOS methods (AP, PM, and UCB) and one uniformly random operator selection strategy. The HV_{avg} and HV_{std} over 25 independent runs achieved by each algorithm on each problem

Problem	D	AOS-LSMOEA(AP)	AOS-LSMOEA(PM)	AOS-LSMOEA(UCB)	UR-LSMOEA
TREE1	3000	0.8175 (0.0097)	0.8127 (0.0136)	0.8100 (0.0114)	0.8051 (0.0118)
TREE2	3000	0.8451 (0.0064)	0.8455 (0.0053)	0.8360 (0.0094)	0.8359 (0.0107)
TREE3	6000	0.8787 (0.0115)	0.8789 (0.0079)	0.8698 (0.0117)	0.8697 (0.0104)
TREE4	6000	0.8815 (0.1372)	0.8997 (0.0768)	0.8547 (0.1118)	0.8514 (0.1384)
TREE5	6000	0.9162 (0.0133)	0.9179 (0.0123)	0.8975 (0.0269)	0.9072 (0.0159)
TREE6	30,000	0.9165 (0.0051)	0.9131 (0.0096)	0.9079 (0.0092)	0.9098 (0.0080)
TREE7	30,000	0.8213 (0.0136)	0.8101 (0.0664)	0.7357 (0.1479)	0.7468 (0.1273)
$+/-/\approx$			0/0/7	6/0/1	5/0/2

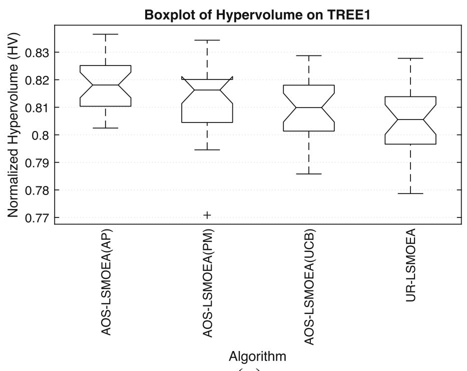
algorithm on each problem is plotted. The Wilcoxon rank-sum test [14] is used to compare the results obtained by the AOS-LSMOEA(AP) and other compared algorithms with a significance level of 0.05. The "$+$", "$-$" and "$\approx$" signs summarize the Wilcoxon rank-sum test results, i.e., the number of problems on which the performance of AOS-LSMOEA(AP) is significantly better than, worse than, or almost similar to that of the corresponding algorithm, respectively. All the investigated algorithms are implemented in PlatEMO [41] (version 2.9) with MATLAB 2019b. The experiments were run on the same computing platform, i.e., Dell Precision T3630 with Intel Core i7-8700 processor and 32 GB RAM.

4.2 Experimental results

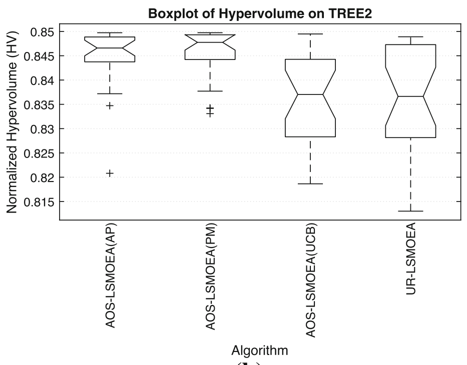
4.2.1 Different operator selection strategies

Three variants of AOS-LSMOEA are compared to determine which AOS method among PM, AP, and UCB is the best for solving TREE problems. Additionally, AOS-

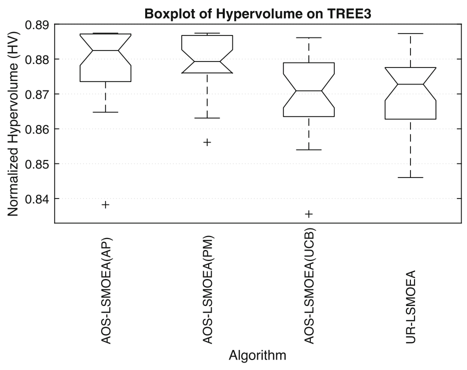
Fig. 3 Comparison of AOS-LSMOEA(AP), AOS-LSMOEA(UR,AP) and AOS-LSMOEA(AP,UR). The HV$_{avg}$ and HV$_{std}$ over 25 independent runs achieved by each algorithm on each problem



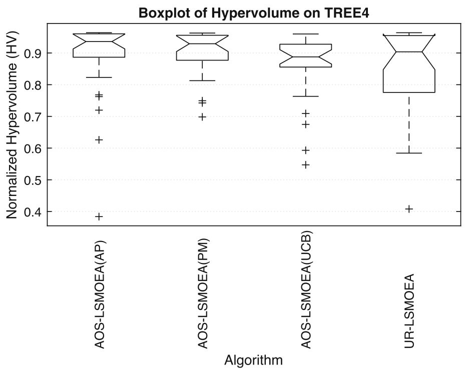
(a)



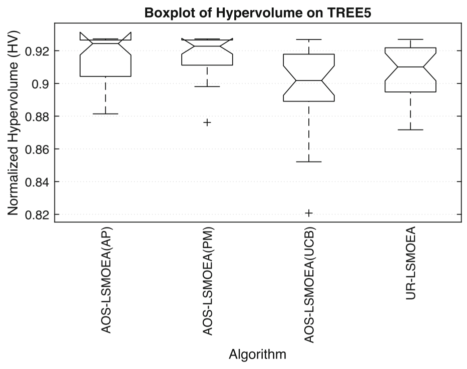
(b)



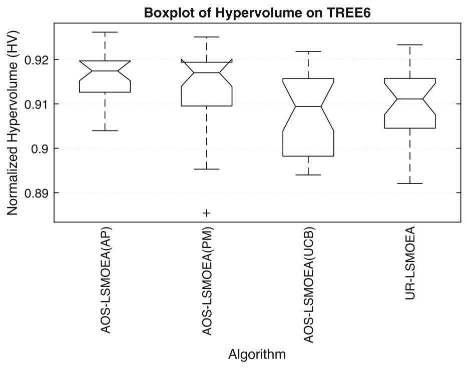
(c)



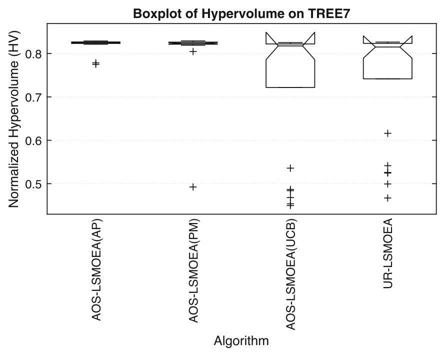
(d)



(e)



(f)



(g)

Table 4 Comparison of AOS-LSMOEA(AP), AOS-LSMOEA(UR,AP) and AOS-LSMOEA(AP,UR). The HV$_{avg}$ and HV$_{std}$ over 25 independent runs achieved by each algorithm on each problem

Problem	*D*	AOS-LSMOEA(AP)	AOS-LSMOEA(UR,AP)	AOS-LSMOEA(AP,UR)
TREE1	3000	0.8175 (0.0097)	0.8053 (0.0114)	0.8114 (0.0094)
TREE2	3000	0.8451 (0.0064)	0.8426 (0.0081)	0.8432 (0.0074)
TREE3	6000	0.8787 (0.0115)	0.8788 (0.0088)	0.8780 (0.0148)
TREE4	6000	0.8815 (0.1372)	0.8968 (0.1409)	0.9062 (0.0638)
TREE5	6000	0.9162 (0.0133)	0.9129 (0.0127)	0.9137 (0.0145)
TREE6	30,000	0.9165 (0.0051)	0.9112 (0.0071)	0.9130 (0.0124)
TREE7	30,000	0.8213 (0.0136)	0.8050 (0.0809)	0.7587 (0.1387)
$+/-/\approx$			0/0/7	0/0/7

Table 5 Comparison of LSMOEA-AP and five popular MOEAs, including iLSMOA, LMOCSO, DGEA, NSGA-II, and MOEA/D-DE. The HV$_{avg}$ and HV$_{std}$ over 25 independent runs achieved by each algorithm on each problem

Problem	AOS-LSMOEA(AP)	iLSMOA	LMOCSO	DGEA	NSGA-II	MOEA/D-DE
TREE1	0.8175 (0.0097)	0.6035 (0.0333)	0.6756 (0.0223)	0.7679 (0.0887)	0.5190 (0.0061)	0.7404 (0.0075)
TREE2	0.8451 (0.0064)	0.6721 (0.0134)	0.7226 (0.0160)	0.8355 (0.0131)	0.6092 (0.0048)	0.7697 (0.0050)
TREE3	0.8787 (0.0115)	0.5282 (0.0264)	0.5642 (0.0359)	0.7978 (0.0842)	0.3425 (0.0067)	0.7180 (0.0116)
TREE4	0.8815 (0.1372)	0.0874 (0.0008)	0.0000 (0.0000)	0.4152 (0.3612)	0.0000 (0.0000)	0.0636 (0.0458)
TREE5	0.9162 (0.0133)	0.4497 (0.0358)	0.5097 (0.0427)	0.6649 (0.1821)	0.1680 (0.0075)	0.6727 (0.0136)
TREE6	0.9165 (0.0051)	0.1870 (0.0477)	0.2889 (0.0571)	0.8755 (0.0776)	0.0000 (0.0000)	0.5773 (0.0240)
TREE7	0.8213 (0.0136)	0.5732 (0.0214)	0.6120 (0.0306)	0.7873 (0.0199)	0.3882 (0.0018)	0.7401 (0.0065)
$+/-/\approx$		7/0/0	7//0/0	5/0/2	7/0/0	7/0/0

LSMOEA is compared with its counterpart using a uniformly random operator selection strategy (denoted as UR-LSMOEA) and the same candidate operators to assess the effectiveness of AOS used in our algorithm. The HV$_{avg}$ and HV$_{std}$ of AOS-LSMOEA(AP), AOS-LSMOEA(PM), AOS-LSMOEA(UCB), and UR-LSMOEA are listed in Table 3. The box plots of HV values are displayed in Fig. 3.

According to the statistical tests, AOS-LSMOEA(AP) performs significantly better than AOS-LSMOEA(UCB) on six out of seven TREE problems. These two algorithms perform similarly on the remaining problem. AOS-LSMOEA(AP) also performs significantly better than UR-LSMOEA on five out of seven TREE problems, and the compared algorithms get similar performance on the remaining two problems. Besides, AOS-LSMOEA(AP) has a similar performance to AOS-LSMOEA(PM) on all the seven problems. When looking at the box plots of HV values in Fig. 3, it can be observed that AOS-LSMOEA(AP) has achieved a better median HV value than the other three algorithms on almost all the test problems except for TREE2. Thus, it can be concluded that the AP method is the best one among PM, AP, and UCB for solving the TREE problem. To investigate the contribution of AOS on offspring generation and environmental selection operators, as well as to further demonstrate the superiority of AOS over uniform random selection strategy, we compared AP with two mixed selection strategies. The results achieved by AOS-LSMOEA(AP), AOS-LSMOEA(UR,AP), and AOS-LSMOEA(AP,UR) are listed in Table 4. Though the three compared algorithms have performed similarly, AOS-LSMOEA(AP) has achieved better HV$_{avg}$ than AOS-LSMOEA(UR,AP) and AOS-LSMOEA(AP,UR), except for TREE4. The result also indicates the advantages of using AP for solving TREE problems.

4.2.2 AOS-LSMOEA(AP) versus some popular MOEAs

Several powerful and popular MOEAs, including iLSMOA, LMOCSO, DGEA, NSAG-II, MOEA/D-DE, are compared with the proposed AOS-LSMOEA(AP). The comparison results are given in Table 5. In this table, the AOS-LSMOEA(AP) obtains the maximal HV$_{avg}$ on all the seven TREE problems. According to the statistical test results, AOS-LSMOEA(AP) performs significantly better than iLSMOA, LMOCSO, NSGA-II, and MOEA/D-DE on all seven problems. AOS-LSMOEA(AP) performs better than DGEA on five problems, and it achieves a similar performance to DGEA on the other two problems. Experimental results also show that UR-LSMOEA achieves better

Table 6 Comparison of AOS-LSMOEA(AP) and 15 hybrid algorithms that combine different offspring generation and environment selection operators. The HV$_{avg}$ and HV$_{std}$ over 25 independent runs achieved by each algorithm on each problem

Problem	AOS-LSMOEA(AP)	HA(OG1,ES1)	HA(OG2,ES1)	HA(OG3,ES1)	HA(OG4,ES1)	HA(OG5,ES1)	HA(OG1,ES2)	HA(OG2,ES2)
TREE1	0.8175 (0.0097)	0.8195 (0.0099)	0.6469 (0.0192)	0.5190 (0.0061)	0.6368 (0.0207)	0.5651 (0.1094)	0.8217 (0.0070)	0.6498 (0.0206)
TREE2	0.8451 (0.0064)	0.8477 (0.0024)	0.7050 (0.0157)	0.6092 (0.0048)	0.6946 (0.0143)	0.5683 (0.1478)	0.8442 (0.0060)	0.7053 (0.0163)
TREE3	0.8787 (0.0115)	0.8841 (0.0051)	0.5318 (0.0415)	0.3425 (0.0067)	0.5062 (0.0349)	0.4115 (0.1416)	0.8843 (0.0048)	0.5382 (0.0380)
TREE4	0.8815 (0.1372)	0.9506 (0.0180)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)	0.0878 (0.0008)	0.9075 (0.0617)	0.0000 (0.0000)
TREE5	0.9162 (0.0133)	0.9247 (0.0041)	0.4863 (0.0430)	0.1680 (0.0075)	0.4466 (0.0383)	0.3399 (0.1333)	0.9163 (0.0203)	0.4881 (0.0481)
TREE6	0.9165 (0.0051)	0.9184 (0.0049)	0.2849 (0.0818)	0.0000 (0.0000)	0.2266 (0.0734)	0.1536 (0.0627)	0.9135 (0.0077)	0.2668 (0.0721)
TREE7	0.8213 (0.0136)	0.3531 (0.1076)	0.5998 (0.0352)	0.3882 (0.0018)	0.5827 (0.0463)	0.4514 (0.1964)	0.8251 (0.0026)	0.5983 (0.0310)
$+/-/\approx$		1/3/3	7//0/0	7/0/0	7//0/0	7//0/0	1/1/5	7//0/0

Problem	HA(OG3,ES2)	HA(OG4,ES2)	HA(OG5,ES2)	HA(OG1,ES3)	HA(OG2,ES3)	HA(OG3,ES3)	HA(OG4,ES3)	HA(OG5,ES3)
TREE1	0.6308 (0.0198)	0.6367 (0.0214)	0.6037 (0.0341)	0.8151 (0.0078)	0.6524 (0.0210)	0.6336 (0.0171)	0.6375 (0.0204)	0.6035 (0.0333)
TREE2	0.6890 (0.0153)	0.6946 (0.0141)	0.6723 (0.0132)	0.8485 (0.0009)	0.7052 (0.0167)	0.6893 (0.0155)	0.6949 (0.0142)	0.6721 (0.0134)
TREE3	0.5082 (0.0272)	0.5017 (0.0333)	0.5287 (0.0277)	0.8840 (0.0049)	0.5231 (0.0452)	0.5070 (0.0244)	0.5067 (0.0381)	0.5282 (0.0264)
TREE4	0.0000 (0.0000)	0.0000 (0.0000)	0.0878 (0.0013)	0.9463 (0.0391)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)	0.0874 (0.0008)
TREE5	0.4651 (0.0302)	0.4539 (0.0433)	0.4462 (0.0294)	0.9250 (0.0026)	0.4913 (0.0453)	0.4617 (0.0318)	0.4549 (0.0435)	0.4497 (0.0358)
TREE6	0.2306 (0.0557)	0.2263 (0.0687)	0.1894 (0.0476)	0.9163 (0.0075)	0.2713 (0.0665)	0.2322 (0.0591)	0.2302 (0.0720)	0.1870 (0.0477)
TREE7	0.5746 (0.0269)	0.5863 (0.0468)	0.5746 (0.0211)	0.8254 (0.0019)	0.6023 (0.0244)	0.5766 (0.0258)	0.5864 (0.0466)	0.5732 (0.0214)
$+/-/\approx$	7//0/0	7//0/0	7//0/0	0/2/5	7//0/0	7//0/0	7//0/0	7//0/0

Table 7 The average run-time (RT_{avg}) over 25 independent runs achieved by each algorithm on each problem (Unit: Second)

Problem	D	iLSMOA	LMOCSO	DGEA	NSGA-II	MOEA/D-DE	UR-LSMOEA	AOS-LSMOEA(AP)
TREE1	3000	17.04	37.31	35.40	31.82	17.29	24.36	24.57
TREE2	3000	26.35	78.06	69.36	59.75	28.85	35.63	33.73
TREE3	6000	60.23	108.57	147.42	122.80	69.69	70.89	73.27
TREE4	6000	60.85	111.71	147.88	121.52	68.37	70.11	70.58
TREE5	6000	68.93	130.12	168.80	137.24	74.29	79.25	79.27
TREE6	30,000	1340.32	2759.12	4190.29	3430.34	1595.48	1639.41	1411.82
TREE7	30,000	1296.68	2694.91	3943.40	3412.44	1584.93	1633.94	1430.01
Sum of RT_{avg}		2870.39	5919.80	8702.55	7315.91	3438.90	3553.59	3123.24

results than iLSMOA, LMOCSO, DGEA, NSGA-II, and MOEA/D-DE, and the proposed AOS-LSMOEA(AP) performs better than UR-LSMOEA. Generally, NSGA-II and iLSMOA have performed poorly, which may be attributed to their disadvantage in diversity maintenance. By contrast, MOEA/D-DE paid more attention to diversity maintenance, leading to slight degeneration in convergence enhancement. DGEA and LMOCSO are good at balancing convergence and diversity, but there is no adaptive strategy to adjust the balance along with the evolution, leading to their failure on TREE4. With the adaptive strategy and involvement of both convergence enhancement and diversity maintenance operations, AOS-LSMOEA has achieved the most competitive results on almost all the test instances, indicating the effectiveness of the adaptive operator selection method.

4.2.3 AOS-LSMOEA(AP) versus different variants

In the following, AOS-LSMOEA(AP) is compared with fifteen hybrid algorithms using different combinations of offspring generation and environmental selection operators to validate AOS's effectiveness comprehensively. The comparison results are listed in Table 6. According to the statistical tests, AOS-LSMOEA(AP) statistically performs better than 12 out of 15 different combinations on seven TREE problems. AOS-LSMOEA(AP) has a similar performance with HV(OG1,ES2), but it performs slightly worse than HV(OG1,ES1) and HV(OG1,ES3). It can be stated that AOS-LSMOEA(AP) has competitive performance with the best combination of the used offspring generation and environmental selection operators.

The results show that the hybrid algorithm HV(OG1,ES1) is the best combination of offspring generation and environmental selection operators among all the possible combinations. Though the best combination HV(OG1,ES1) slightly performs better than AOS-LSMOEA(AP), it has bad result on TREE7. By contrast, AOS-LSMOEA(AP) consistently achieves good results on all the test problems. Thus, AOS-LSMOEA(AP) performance is more robust than the human-made hybrid algorithm that combines different offspring generation and environmental selection strategies. Moreover, to achieve a high-performance human-made hybrid algorithm, one usually needs to test many combinations to select the best one, which generally requires high computational costs.

4.2.4 Run-time analysis

Furthermore, the average run-time (RT_{avg}) and standard deviation of run-time (RT_{std}) over 25 independent runs achieved by each problem are recorded and listed in Table 7. AOS-LSMOEA(AP) does not increase computing time significantly. On each problem, the sum of RT_{avg} of AOS-LSMOEA(AP) is only longer than that of iLSMOA and MOEA/D-DE, and it is shorter than the other five algorithms. Thus, the additional computational cost brought by AP in AOS-LSMOEA(AP) is negligible. It can be stated that the AOS-LSMOEA(AP) significantly improves the performance of iLSMOA on TREE problems. At the same time, the computational cost is only increased slightly. Therefore, the proposed AOS-LSMOEA(AP) is an effective and efficient algorithm for solving TREE problems.

Moreover, the average frequency of the offspring generation operators and environmental selection operators selected by AOS and applied in the search process are shown in Fig. 4. According to the histogram, it can be found that the "LSMOF" (offspring generation operator from LSMOF algorithm) usually performs well on the TREE problems, and it is the most frequently used offspring generation operator on all the test TREE problems. For the environmental selection process, operators from NSGA-II, IBEA, and SPEA2 are most frequently used in TREE1 to TREE6, whereas NSGA-II has minimal application frequency on TREE7. This result indicates that the environmental selection operator in NSGA-II is not suitable for the TREE7 problem, which is also validated by the poor results obtained by HA (combination of LSMOF and NSGA-II) on TREE7. Thanks to AP used to select the suitable operator, other qualified operators

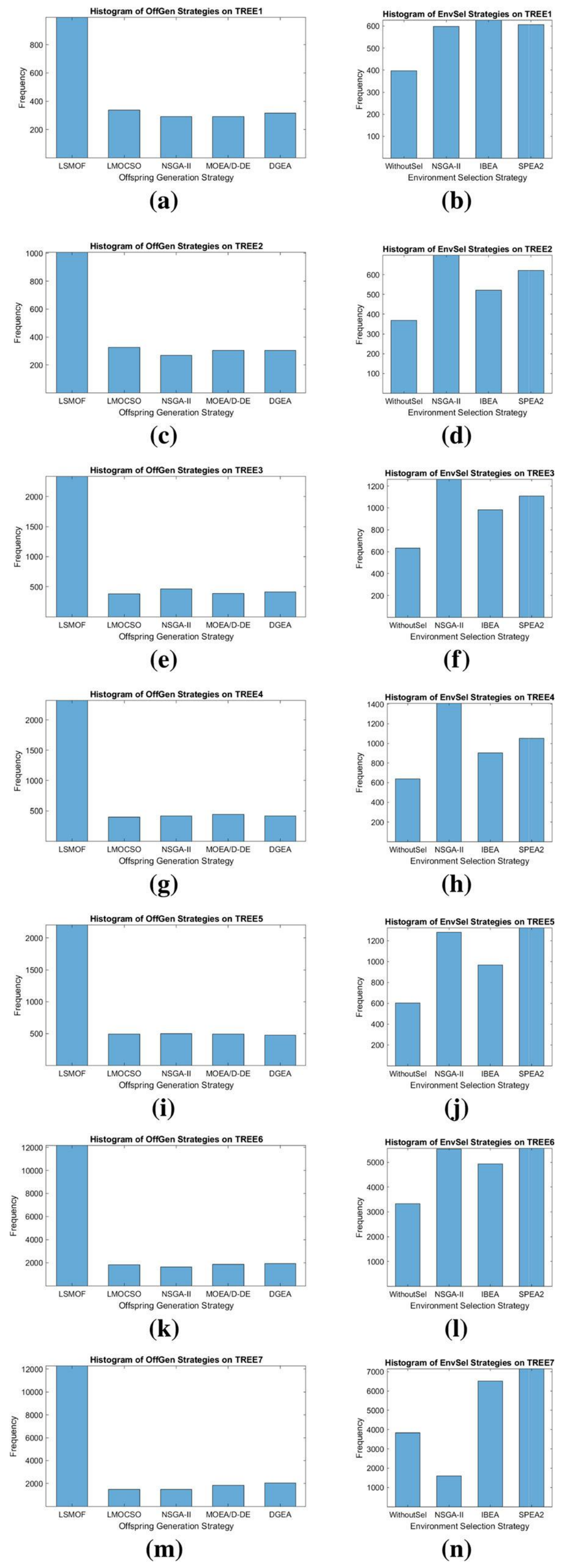


Fig. 4 Histogram of operator selection on TREE problems from AOS-LSMOEA(AP)

rather than NSGA-II are selected in AOS on the environmental selection operator. This result also shows the superiority of combining AOS and existing LSMOEAs in solving TREE problems.

5 Conclusion

This paper proposed an adaptive LSMOEA framework with adaptive operator selection (AOS), termed AOS-LSMOEA, to solve real-world LSMOPs, i.e., the TREE problems. In the proposed AOS-LSMOEA, multiple offspring generation and environmental selection operators are included, and the AOS method automatically selects suitable operators to use in each iteration. Experimental studies show that AOS-LSMOEA performs better than iLSMOA, LMOCSO, DGEA, NSAG-II, MOEA/D-DE and UR-LSMOEA. The superiority of AOS-LSMOEA can be attributed to two aspects. On the one hand, by including multiple operators in the algorithm, its search ability can be enhanced since different operators have different strengths during the search. On the other hand, adopting AOS can identify and select proper operators from available operators. Besides, the histograms of operator selection on different TREE problems reveal that the offspring generation strategy of LSMOF is mostly used. While, for the environmental selection operator, the trend of application frequency is changed on TREE7. Fortunately, AOS has identified and appropriately addressed this sudden change, and thus suitable operator can be selected. This work also shows that it is promising to use automatic algorithm configuration methods, such as adaptive operator selection and automatic parameter tuning, to design new MOEAs or LSMOEAs to solve challenging MOPs in real-world applications.

References

1. Antonio LM, Coello CAC (2013) Use of cooperative coevolution for solving large scale multiobjective optimization problems. In: Proceedings of 2013 IEEE congress on evolutionary computation (CEC 2013), pp 2758–2765
2. Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. Mach Learn 47(2):235–256
3. Beume N, Naujoks B, Emmerich M (2007) SMS-EMOA: multiobjective selection based on dominated hypervolume. Eur J Oper Res 181(3):1653–1669
4. Calderín JF, Masegosa AD, Pelta DA (2016) An algorithm portfolio for the dynamic maximal covering location problem. Memet Comput 9(2):141–151
5. Cheng R (2016) Nature inspired optimization of large problems. Ph.D. thesis, University of Surrey
6. Cheng R, Jin Y, Olhofer M, Sendhoff B (2017) Test problems for large-scale multiobjective and many-objective optimization. IEEE Trans Cybern 47(12):4108–4121

7. Coello CAC, Lamont GB, Van Veldhuizen DA et al (2007) Evolutionary algorithms for solving multi-objective problems, vol 5. Springer, Berlin
8. Coello CAC, Sierra MR (2004) A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In: Proceedings of Mexican international conference on artificial intelligence (MICAI). Springer, pp 688–697
9. Consoli PA, Mei Y, Minku LL, Yao X (2016) Dynamic selection of evolutionary operators based on online learning and fitness landscape analysis. Soft Comput 20(10):3889–3914
10. Corne DW, Jerram NR, Knowles JD, Oates MJ (2001) PESA-II: region-based selection in evolutionary multi-objective optimization. In: Proceedings of the 3rd annual conference on genetic and evolutionary computation, GECCO'01. Morgan Kaufmann Publishers Inc., pp 283-290
11. Deb K, Jain H (2014) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. IEEE Trans Evol Comput 18(4):577–601
12. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
13. Deb K, Thiele L, Laumanns M, Zitzler E (2005) Scalable test problems for evolutionary multiobjective optimization. Evolutionary multiobjective optimization. Advanced Information and Knowledge Processing. Springer, London, pp 105–145
14. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18
15. Fialho Á, Costa LD, Schoenauer M, Sebag M (2008) Extreme value based adaptive operator selection. In: Rudolph G, Jansen T, Lucas SM, Poloni C, Beume N (eds) Parallel problem solving from nature—PPSN X, vol 5199. Springer, Berlin, pp 175–184
16. Fialho Á, Costa LD, Schoenauer M, Sebag M (2010) Analyzing bandit-based adaptive operator selection mechanisms. Ann Math Artif Intell 60(1–2):25–64
17. Goldberg DE (1990) Probability matching, the magnitude of reinforcement, and classifier system bidding. Mach Learn 5:407–425
18. Gonçalves RA, Almeida CP, Pozo A (2015) Upper confidence bound (UCB) algorithms for adaptive operator selection in MOEA/D. In: International conference on evolutionary multi-criterion optimization. Springer, pp 411–425
19. He C, Cheng R, Tian Y, Zhang X (2020) Iterated problem reformulation for evolutionary large-scale multiobjective optimization. In: Proceedings of 2020 IEEE congress on evolutionary computation (CEC 2020). IEEE, pp 1–8
20. He C, Cheng R, Yazdani D (2020) Adaptive offspring generation for evolutionary large-scale multiobjective optimization. IEEE Trans Syst Man Cybern Syst 52(2):786–798
21. He C, Cheng R, Zhang C, Tian Y, Chen Q, Yao X (2020) Evolutionary large-scale multiobjective optimization for ratio error estimation of voltage transformers. IEEE Trans Evol Comput 24(5):868–881
22. He C, Huang S, Cheng R, Tan KC, Jin Y (2021) Evolutionary multiobjective optimization driven by generative adversarial networks (GANs). IEEE Trans Cybern 51(6):3129–3142
23. He C, Li L, Tian Y, Zhang X, Cheng R, Jin Y, Yao X (2019) Accelerating large-scale multiobjective optimization via problem reformulation. IEEE Trans Evol Comput 23(6):949–961
24. He C, Tian Y, Jin Y, Zhang X, Pan L (2017) A radial space division based many-objective optimization evolutionary algorithm. Appl Soft Comput 61:603–621
25. Huband S, Hingston P, Barone L, While L (2006) A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans Evol Comput 10(5):477–506
26. Ishibuchi H, Murata T (1998) Multiobjective genetic local search algorithm and its application to flowshop scheduling. IEEE Trans Syst Man Cybern Part C 28(3):392–403
27. Kukkonen S, Lampinen J (2005) GDE3: the third evolution step of generalized differential evolution. In: Proceedings of 2005 IEEE congress on evolutionary computation (CEC 2005). IEEE, pp 443–450
28. Li H, Zhang Q (2009) Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. IEEE Trans Evol Comput 13:284–302
29. Li K, Fialho A, Kwong S, Zhang Q (2014) Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 18(1):114–130
30. Li L, Wang X (2021) An adaptive multiobjective evolutionary algorithm based on grid subspaces. Memet Comput 13(2):249–269
31. Liu HL, Gu F, Zhang Q (2014) Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. IEEE Trans Evol Comput 18(3):450–455
32. Ma X, Liu F, Qi Y, Wang X, Li L, Jiao L, Yin M, Gong M (2015) A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. IEEE Trans Evol Comput 20(2):275–298
33. Nebro AJ, Durillo JJ, Garcia-Nieto J, Coello CC, Luna F, Alba E (2009) SMPSO: a new PSO-based metaheuristic for multi-objective optimization. In: 2009 IEEE symposium on computational intelligence in multi-criteria decision-making (MCDM). IEEE, pp 66–73
34. Omidvar MN, Yang M, Mei Y, Li X, Yao X (2017) DG2: a faster and more accurate differential grouping for large-scale black-box optimization. IEEE Trans Evol Comput 21(6):929–942
35. Pan L, He C, Tian Y, Wang H, Zhang X, Jin Y (2018) A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization. IEEE Trans Evol Comput 23(1):74–88
36. Ponsich A, Jaimes AL, Coello CAC (2013) A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications. IEEE Trans Evol Comput 17(3):321–344
37. Potter MA, De Jong KA (1994) A cooperative coevolutionary approach to function optimization. In: Davidor Y, Schwefel HP, Männer R (eds) Parallel problem solving from nature—PPSN III. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 249–257
38. Qin S, Sun C, Jin Y, Tan Y, Fieldsend J (2021) Large-scale evolutionary multiobjective optimization assisted by directed sampling. IEEE Trans Evol Comput 25(4):724–738
39. Thierens D (2005) An adaptive pursuit strategy for allocating operator probabilities. In: Proceedings of the 7th annual conference on genetic and evolutionary computation, GECCO'05. Association for Computing Machinery, pp 1539–1546
40. Thierens D (2007) Adaptive strategies for operator allocation. In: Lobo FG, Lima CF, Michalewicz Z (eds) Parameter setting in evolutionary algorithms. Studies in computational intelligence, vol 54. Springer, Berlin, pp 77–90
41. Tian Y, Cheng R, Zhang X, Jin Y (2017) PlatEMO: a matlab platform for evolutionary multi-objective optimization [educational forum]. IEEE Comput Intell Mag 12(4):73–87
42. Tian Y, Si L, Zhang X, Cheng R, He C, Tan KC, Jin Y (2021) Evolutionary large-scale multi-objective optimization: a survey. ACM Comput Surv (CSUR) 54(8):1–34
43. Tian Y, Zheng X, Zhang X, Jin Y (2020) Efficient large-scale multiobjective optimization based on a competitive swarm optimizer. IEEE Trans Cybern 50(8):3696–3708
44. Wang Y, Li B (2009) Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. Memet Comput 2(1):3–24

45. Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
46. Zhang X, Tian Y, Cheng R, Jin Y (2016) A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. IEEE Trans Evol Comput 22(1):97–112
47. Zille H, Ishibuchi H, Mostaghim S, Nojima Y (2018) A framework for large-scale multiobjective optimization based on problem transformation. IEEE Trans Evol Comput 22(2):260–275
48. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195
49. Zitzler E, Künzli S (2004) Indicator-based selection in multi-objective search. In: Yao X, Burke EK, Lozano JA, Smith J, Merelo-Guervós JJ, Bullinaria JA, Rowe JE, Tiňo P, Kabán A, Schwefel HP (eds) Parallel problem solving from nature—PPSN VIII. Springer, Berlin, pp 832–842
50. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 3(4):257–271
51. Ziztler E, Laumanns M, Thiele L (2002) SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. Evolutionary methods for design, optimization, and control, pp 95–100