mathematics

MDPI

Article

Self-Adaptive Constrained Multi-Objective Differential Evolution Algorithm Based on the State–Action–Reward–State–Action Method

Qingqing Liu, Caixia Cui and Qinqin Fan *

Logistics Research Center, Shanghai Maritime University, Shanghai 201306, China; 202030510155@stu.shmtu.edu.cn (Q.L.); cuicaixia0928@163.com (C.C.)
* Correspondence: qqfan@shmtu.edu.cn; Tel.: +86-135-2446-0557

Abstract: The performance of constrained multi-objective differential evolution algorithms (CMOEAs) is mainly determined by constraint handling techniques (CHTs) and their generation strategies. To realize the adaptive adjustment of CHTs and generation strategies, an adaptive constrained multi-objective differential evolution algorithm based on the state–action–reward–state–action (SARSA) approach (ACMODE) is introduced in the current study. In the proposed algorithm, the suitable CHT and the appropriate generation strategy can be automatically selected via a SARSA method. The performance of the proposed algorithm is compared with four other famous CMOEAs on five test suites. Experimental results show that the overall performance of the ACMODE is the best among all competitors, and the proposed algorithm is capable of selecting an appropriate CHT and a suitable generation strategy to solve a particular type of constrained multi-objective optimization problems.

Keywords: constrained multi-objective optimization; evolutionary computation; reinforcement learning; SARSA method

MSC: 68Txx

Citation: Liu, Q.; Cui, C.; Fan, Q. Self-Adaptive Constrained Multi-Objective Differential Evolution Algorithm Based on the State–Action–Reward–State–Action Method. *Mathematics* **2022**, *10*, 813. https://doi.org/10.3390/math10050813

Academic Editors: Linqiang Pan, Zhihua Cui, Harish Garg, Thomas Hanne and Gai-Ge Wang

Received: 22 January 2022
Accepted: 2 March 2022
Published: 3 March 2022

1. Introduction

Constrained multi-objective optimization problems (CMOPs) are commonly found in the field of engineering optimization, such as robot's design optimization [1], compressed-air station scheduling problem [2] and scheduling optimization of microgrid [3]. To effectively solve CMOPs, various improved CMOEAs have been proposed. For example, Wang et al. [4] proposed a cooperative multi-objective evolutionary algorithm with a propulsive population (CMOEA-PP) to achieve a tradeoff among the diversity, the convergence, and the feasibility in different evolutionary stages. Datta et al. [5] combined the evolutionary multi-objective optimization method with the penalty function method, and proposed a bi-objective hybrid constrained optimization algorithm (HyCon) to deal with CMOPs. Yuan et al. [6] proposed an indicator-based evolutionary algorithm to prevent the population from falling into local areas. Cui et al. [7] proposed an adaptive constraint handling technique (CHT), which can adaptively select suitable CHT from three state-of-the-art CHTs via the Q-learning method.

Although various CMOEAs have been proposed to carry out adaptation selection of CHTs, their search strategies are generally constant during the entire evolutionary process. Therefore, it may not be effective when solving different types of CMOPs. To alleviate the above issues, an adaptive constraint multi-objective differential evolution algorithm based on SARSA method (named as ACMODE) is proposed. In the ACMODE, three commonly used CHTs are selected and the SARSA [8] is utilized to select appropriate CHTs during different stages of the evolution. Moreover, two-generation strategies are chosen in differential evolution (DE), and the SARSA method is used to select appropriate generation

strategies in the next iteration. Simulation experiments with four other CMOEAs are carried out on five constrained multi-objective test suites, and the results show that the ACMODE is more competitive than the four other CMOEAs.

The main contribution of this work is to realize the adaptive adjustment of CHTs and generation strategies in the ACMODE. Three CHTs and two DE's generation strategies are integrated into the proposed algorithm. The simulation results show that the appropriate CHT and generation strategy can be self-adaptively selected from CHT and generation strategy pools to solve a particular type of CMOPs during the entire evolutionary process. Therefore, ACMODE can integrate the advantages of different CHTs and generation strategies when solving different CMOPs.

The rest of this paper is arranged as follows. Section 2 reviews the related works. Section 3 briefly introduces some basic concepts. The details of the proposed algorithm are presented in Section 4. Subsequently, the experimental results and analyses are shown in Section 5. Some discussions are provided in Section 6. Section 7 draws some conclusions.

2. Literature Review

2.1. Constrained Multi-Objective Evolutionary Algorithms

Recently, a large number of CMOEAs have been proposed to solve CMOPs. For example, Yu et al. [9] proposed a corner point-based algorithm, which has two stages: in the first stage, corner points are quickly found, and in the second stage, a new diversity and convergence strategy is used to approach the real Pareto front. In [10], CMOPs can be divided into three types according to the relationship between the constrained Pareto-optimal front and unconstrained Pareto front. By considering potential problem types, a new CHT was proposed to solve CMOPs. Fan et al. [11] introduced a novel framework, in which the entire search process is divided into push stage and pull stage. Constraints have not been considered in the push stage, while an improved epsilon-constraint method is used in the pull phase. To improve the search performance of MOEAs, a local search mechanism [12] was proposed, which can contain constraint information and does not need to explicitly calculate gradient information. Liu et al. [13] developed an indicator-based CMOEA framework, in which indicator-based MOEAs and CHTs are effectively combined to solve CMOPs. In [14], a coevolutionary framework was proposed to solve CMOPs via using two different populations. A two-phase framework (ToP) was proposed in Ref. [15]. In the ToP, a CMOP was transformed into constrained single-objective optimization problems for locating promising regions in the first phase, and a specific and efficient CMOEA was employed to find feasible solutions in the second stage. In addition, a dual-stage dual-population evolutionary algorithm [16] was proposed recently. The whole search process was divided into exploration and exploitation, and two populations evolved with and without considering the constraints. Based on the DE [17], a new DE variant named IMDE [18] was proposed, which used infeasible solutions to guide mutation operators and applied multiple combinations of mutation strategies and control parameters to enhance the search performance. Yu et al. [19] proposed a dynamic selection preference-assisted constrained multiobjective differential evolutionary algorithm (DSPCMDE). In DSPCMDE, the selection preference of each individual changes from objective functions to constraint function as the evolutionary process. To balance feasibility, convergence and distribution, Yang et al. [20] proposed a multi objective differential evolutionary algorithm based on partition selection (MODE-PS). In MODE-PS, a CMOP is divided into several subproblems by objective space to keep the distribution. Each subspace saves one feasible solution to a feasible solution set to maintain the feasibility of the subspaces. Once there are feasible solutions in one subspace, the individual selection strategy is changed from constraint search to non-constraint search to accelerate the convergence. Lin et al. [21] proposed a multi-objective differential evolution with dynamic hybrid constraint handling mechanism (MODE-DCH) to tackle CMOPs. In the MODE-DCH, the different search models combined with different CHTs are used. In addition, Xiao et al. [22] proposed a new mutation mechanism. In this mechanism, DE mutation operator and Gaussian mutation

operator are used to deal with infeasible solutions and feasible solutions respectively. Wang et al. [23] proposed a cooperative differential evolution framework (CCMODE) in which the mutation operator is used to guide the infeasible individuals to move to the feasible region.

2.2. Self-Adaptive Evolutionary Algorithms

No strategy can perform best on all types of CMOPs due to No Free Lunch [24]. To integrate the advantage of different strategies, many self-adaptive constrained multi-objective evolutionary algorithms have been introduced. Based on the improved epsilon-constraint method, Yang et al. [25] proposed a multi-objective differential evolutionary algorithm named MODE-SaE. In MODE-SaE, the global search and local search can be self-adaptively adjusted by self-switching parameters of the search engine to balance the convergence and distribution. In [26], a CMOP was decomposed into multiple subproblems. Each subproblem has a subpopulation in a subregion. The appropriate CHT can be adaptively selected in each subregion. Based on decomposition and the DE, Liu and Bi [27] presented an adaptive ε-constraint MOEA to make full use of the information of infeasible solutions. They also proposed an adaptive DE mutation strategy to increase search efficiency and avoid falling into the local optimum. In [28], an adaptive search operator scheme was introduced. When a search operator hits a difficult patch in the search space, the scheme "reacts" to that by potentially calling upon a different search operator. Zhang et al. [29] proposed a constrained multi-objective optimization algorithm based on adaptive ε-truncation (ε-T-CMOA) to further improve the distribution and convergence of the obtained solutions. In [30], an adaptive repair approach was proposed to improve the efficiency of constraint handling in non-dominance. Repairing was carried out on the solutions that dominate all feasible solutions or have the smallest constraint violation.

3. Basic Concepts

3.1. Constrained Multi-Objective Optimization Problem

CMOPs can be mathematically defined as follows:

$$\begin{array}{ll} \min & f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x}))^T \\ s.t. & g_j(\boldsymbol{x}) \leq 0, j = 1, 2, \cdots, p \\ & h_j(\boldsymbol{x}) = 0, j = p+1, \cdots, q \\ & \boldsymbol{x} = (x_1, x_2, \cdots, x_D)^T \in O, \end{array} \tag{1}$$

where $\boldsymbol{x}$ is the D-dimensional decision vector; $f(\boldsymbol{x})$ is an objective vector containing m objectives; $g_j(\boldsymbol{x})$ is the jth inequality constraint and p denotes the number of inequality constraints; $h_j(\boldsymbol{x})$ is the (j-p)th equality constraint and has (q-p) equality constraints; O is the decision space.

The constraint violation degree of the solution $\boldsymbol{x}$ can be computed as:

$$C_j(\boldsymbol{x}) = \begin{cases} \max\left(0, g_j(\boldsymbol{x})\right), & 1 \leq j \leq p \\ \max\left(0, \left|h_j(x) - \delta\right|\right), & p+1 \leq j \leq q \end{cases}, \tag{2}$$

where δ is the tolerance value of equality constraints and is usually set as a small positive value. The overall constraint violation degree of the solution $\boldsymbol{x}$ can be calculated as follows:

$$C(\boldsymbol{x}) = \sum_{j=1}^{q} C(\boldsymbol{x}). \tag{3}$$

For Equation (3), the solution $\boldsymbol{x}$ is a feasible solution when $C(\boldsymbol{x}) = 0$. On the contrary, it is an infeasible solution.

3.2. Concepts in Multi-Objective Optimization Problem

Basic concepts in multi-objective optimization are represented as follows [31]:

Definition 1 (Dominance relation). *If there are two vectors* $\boldsymbol{u}$ *and* $\boldsymbol{v}$ *in the minimization optimization problem,* $\forall n \in \{1, 2, \cdots, m\}$, $u_n \leq v_n$ *and* $\boldsymbol{u} \neq \boldsymbol{v}$, *then is said to dominate* $\boldsymbol{v}$, *denoted as* $\boldsymbol{u} \succ \boldsymbol{v}$.

Definition 2 (Pareto optimal set). *For a solution* $\boldsymbol{x}^* \in R^D$, *if and only if there is no other solution* $\mathbf{x}$ *such that* $\boldsymbol{F}(\boldsymbol{x}) \succ \boldsymbol{F}(\boldsymbol{x}^*)$, *it is called a Pareto optimal solution of a CMOP. All the Pareto optimal solutions form the Pareto set (PS), defined as* $\boldsymbol{X}^*$.

Definition 3 (Pareto front). *The Pareto front can be referred to* $PF = \{F(\boldsymbol{x}^*) | \boldsymbol{x}^* \in \boldsymbol{X}^*\}$.

3.3. Constraint Handling Strategies

The CHTs adopted in our proposed algorithm are self-adaptive penalty (SP), constrained-domination principle (CDP) and adaptive tradeoff model (ATM).

3.3.1. SP

SP [32] is a representative way to solve CMOPs which penalizes the infeasible individual with a penalty function. It has two main components: the distance value and the penalty function. SP can be defined as follows:

$$\begin{aligned} M_n(\boldsymbol{x}) &= l_n(\boldsymbol{x}) + b_n(\boldsymbol{x}), \\ l_n(\boldsymbol{x}) &= \begin{cases} C(\boldsymbol{x}), & \text{if } r_f = 0 \\ \sqrt{\widetilde{f}_n(\boldsymbol{x})^2 + C(\boldsymbol{x})^2}, & \text{otherwise} \end{cases}, \\ b_n(\boldsymbol{x}) &= \left(1 - r_f\right) c_n(\boldsymbol{x}) + r_f e_n(\boldsymbol{x}), \\ c_n(\boldsymbol{x}) &= \begin{cases} 0, & \text{if } r_f = 0 \\ C(\boldsymbol{x}), & \text{otherwise} \end{cases}, \\ e_n(\boldsymbol{x}) &= \begin{cases} 0, & \text{if } \boldsymbol{x} \text{ is a feasible individual} \\ \widetilde{f}_n(\boldsymbol{x}), & \text{otherwise} \end{cases}, \end{aligned} \tag{4}$$

where $l_n(\boldsymbol{x})$ is the n-th distance value; $b_n(\boldsymbol{x})$ is the n-th penalty value; $C(\boldsymbol{x})$ is the degree of constrained violation, $C(\boldsymbol{x}) = \frac{1}{q} \sum_{j=1}^{q} \frac{C_j(\boldsymbol{x})}{C_j^{\max}}$; $\widetilde{f}_n(\boldsymbol{x})$ is the normalized objective function value; r_f is feasible rate of the population; Then, the population is sorted based on the m fitness functions $M_1, M_2 \ldots, M_m$ via the nondominated sorting.

3.3.2. CDP

CDP [33] proposed by Deb is a simple and efficient technology to select individuals, which compares pairwise individuals based on the following rules:

- Any feasible solution is preferred to any infeasible solution.
- For two feasible solutions, the Pareto non-dominant individual is preferred.
- For two infeasible solutions, the one with a smaller degree of constraint violation is preferred.

3.3.3. ATM

Based on the feasibility ratio of the current population, ATM [34] divides the evolutionary process into three situations:

- The infeasible situation: The constraint violation is considered as an additional objective. The nondominated sorting is applied, and then half of the individuals with fewer constraint violations in the first layer are sorted in the offspring population, then deleted from the population. The same operation is performed on the remaining individuals until the number of offspring reaches population size.

- The semi-feasible situation: Similar to SP, ATM uses a new function, which is calculated as follows:

$$M_n(\boldsymbol{x}) = \widetilde{f}_n(\boldsymbol{x}) + \widetilde{C}(\boldsymbol{x}),$$
$$\widetilde{f}_n(\boldsymbol{x}) = \frac{f'_n(\boldsymbol{x}) - \min\limits_{\boldsymbol{x}\in P} f'_n(\boldsymbol{x})}{\max\limits_{\boldsymbol{x}\in P} f'_n(\boldsymbol{x}) - \min\limits_{\boldsymbol{x}\in P} f'_n(\boldsymbol{x})},$$
$$\widetilde{C}(\boldsymbol{x}) = \begin{cases} 0, & \boldsymbol{x} \in P_Y \\ \dfrac{\overline{C}(\boldsymbol{x}) - \min\limits_{\boldsymbol{x}\in P_N} \overline{C}(\boldsymbol{x})}{\max\limits_{\boldsymbol{x}\in P_N} \overline{C}(\boldsymbol{x}) - \min\limits_{\boldsymbol{x}\in P_N} \overline{C}(\boldsymbol{x})}, & \boldsymbol{x} \in P_N \end{cases}, \tag{5}$$
$$f'_n(\boldsymbol{x}) = \begin{cases} f_n(\boldsymbol{x}), & \boldsymbol{x} \in P_Y \\ \max\{\varphi * f_{\min} + (1-\varphi) * f_{\min}\}, & \boldsymbol{x} \in P_N \end{cases},$$

where φ represents the feasible ratio of the last iteration population; P_Y and P_N is the set of feasible and infeasible solutions in P. The m fitness functions $M_1, M_2 \ldots, M_m$ are used to sort the population via the nondominated sorting.

- The feasible situation: Nondominated sorting is used to select individuals.

3.4. Performance Metric

In the present work, two widely used performance metrics are employed: the inverted generational distance (*IGD*) and the hypervolume (*HV*).

3.4.1. *IGD*

IGD [35] is calculated as:

$$IGD(H, PF^*) = \frac{\sqrt{\sum_{z^*\in PF^*} d(z^*, H)^2}}{|PF^*|}, \tag{6}$$

where H represents the *PF* approximation; *PF** is a set of solutions obtained by evolutionary algorithms, which is uniformly distributed along the true *PF*; d (z*, H) is the minimum Euclidean distance between individual z* in *PF** and H, $|PF^*|$ denotes the number of points in *PF**. The algorithm with a smaller *IGD* value has better performance [36]. Generally, *IGD* can simultaneously evaluate the convergence and diversity of *PF*.

3.4.2. *HV*

HV [37] can be defined as follows:

$$HV(H) = L\left(\bigcup_{z\in H}[z_1, z_1^r] \times \cdots \times [z_m, z_m^r]\right), \tag{7}$$

where L is the Lebesgue measure; $z = (z_1, \ldots, z_m)$ represents a solution in H; and $\boldsymbol{z}^r = (z_1^r, \ldots, z_m^r)$ denotes a worst point dominated by all the Pareto optimal solutions. A larger *HV* means a better Pareto front set in both the convergence and the diversity [36].

3.5. Basics of DE

Differential evolution algorithm [38] is a simple and efficient meta-heuristic search algorithm. Its main operator steps are as follows:

3.5.1. Generation Strategy

Two commonly seen generation strategies are as follows:
"DE/rand-to-best/1/bin":

$$\boldsymbol{v}_i^G = \boldsymbol{x}_{r1}^G + rand \cdot (\boldsymbol{x}_{best}^G - \boldsymbol{x}_i^G) + F \cdot (\boldsymbol{x}_{r2}^G - \boldsymbol{x}_{r3}^G), \tag{8}$$

$$u_{ij}^G = \begin{cases} v_{ij}^G, R_j \leq CR \text{ or } j = j_{rand} \\ x_{ij}^G, \text{otherwise} \end{cases}, j = 1, 2, \cdots D \,, \tag{9}$$

"DE/current-to-rand/1":

$$\boldsymbol{v}_i^G = \boldsymbol{x}_i^G + rand \cdot (\boldsymbol{x}_{r3}^G - \boldsymbol{x}_i^G) + F \cdot (\boldsymbol{x}_{r1}^G - \boldsymbol{x}_{r2}^G) \,, \tag{10}$$

where $\boldsymbol{x}_i^G$ is the ith individual in the Gth generation; indices r_1, r_2, and r_3, which are all different from i and are randomly generated from 1 to NP (population size). The scale factor is F, which is used to scale differential vectors. $\boldsymbol{x}_{best}^G$ represents the best individual in the Gth generation. R_j is a random number, which ranges from 0 to 1; CR is crossover probability; j_{rand} is an integer randomly generated within [1, D]. It is worth noting that the binomial crossover is not implemented in "DE/current-to-rand/1". The information of the best individual is employed in "DE/rand-to-best/1/bin", so it is able to enhance the convergence. While the diversity can be maintained in "DE/current-to-rand/1", since other randomly selected individuals are learned.

3.5.2. Selection

After the generation strategy, the selection operation is performed to select the good solutions as the parents for the next generation, which can be defined as follows:

$$\boldsymbol{x}_i^{G+1} = \begin{cases} \boldsymbol{u}_i^G, f(\boldsymbol{u}_i^G) \leq f(\boldsymbol{x}_i^G) \\ \boldsymbol{x}_i^G, \text{otherwise} \end{cases} , \tag{11}$$

where $\boldsymbol{x}_i^{G+1}$ is the selected solution that can be used in next generation.

4. Proposed Algorithm

Different CHTs and generation strategies have significant effects on the performance of CMOEAs. To further improve the performance, an adaptive constrained multi-objective differential evolution algorithm (ACMODE) is proposed in the present study. When solving different types of CMOPs in the ACMODE, suitable CHT and generation strategies can be adaptively selected during the whole evolutionary process.

The main operators in the ACMODE are as follows:

4.1. Adaptive Constraint Handling Technology

Different CHTs are suitable for solving different properties of CMOPs, thus the adaptation of CHTs is proposed in the current work. Three commonly used CHTs (SP [32], CDP [33] and ATM [34]) are selected and the SARSA method is used to realize the adaptation of these three different CHTs. To evaluate the performance of each CHT, an improved *IGD* is given as follows [39]:

$$mIGD(H, \overline{PF}) = \frac{\sqrt{\sum_{\bar{z} \in \overline{PF}} d(\bar{z}, H)^2}}{\left|\overline{PF}\right|}, \tag{12}$$

where $\overline{PF}$ is selected from all achieved PF approximations.

The pseudocode of the proposed adaptive CHT method is described in Algorithm 1. The action space can be defined as $\boldsymbol{AC}$ = [SP, CDP, ATM], the state space can be expressed as $\boldsymbol{SV}$ = [excellent, medium, poor], and the value of reward $\boldsymbol{RC}$ is [1, 0, −1] [40]. The form of the Q-table is shown in Table 1:

Table 1. The form of Q-table.

Q-Table	SP	CDP	ATM
excellent	Q (1,1)	Q (1,2)	Q (1,3)
medium	Q (2,1)	Q (2,2)	Q (2,3)
poor	Q (3,1)	Q (3,2)	Q (3,3)

In lines 1 to 2, according to $\boldsymbol{AC}$, the number of individuals choosing each CHT can be determined. Therefore, *mIGD* value can be calculated by Equation (12). In lines 3 to 6, the maximum *mIGD* value represents the individual choosing this CHT in the "poor" state and its reward is -1; the middle *mIGD* value indicates the state is "medium" and its reward is 0; and the reward of "excellent" CHT is 1. State $s^{'}$ is used to predict action $a^{'}$ and then update the Q-table. Finally, action chain $\boldsymbol{AC}$ is updated.

Algorithm 1: Adaptive Constraint Handling Technique

Input: the state vector $\boldsymbol{SV}$ and the reward chain $\boldsymbol{RC}$
Output: action chain $\boldsymbol{AC}$
1 Determine the number of individuals selecting each CHT via $\boldsymbol{AC}$;
2 Calculate *mIGD* value according to Equation (12).
3 Obtain the $s^{'}$ and r according to *mIGD* value, and update $\boldsymbol{SV}$ and $\boldsymbol{RC}$;
4 Use ε-greedy method to predict individual action $a^{'}$ according to $s^{'}$;
5 Update Q-table: $Q(s,a) = Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$;
6 Update action chain $\boldsymbol{AC}$;

4.2. Adaptive Generation Strategy

Different generation strategies play distinct roles in the search process. "DE/current-to-rand/1" has good exploration and search ability, while "DE/rand-to-best/1/bin" possesses good local search capability and its convergence speed is faster than "DE/rand/1". Consequently, two-generation strategies are applied in the proposed algorithm. The process of adaptive generation strategy is as follows:

Step 1: initialize the state s_0 for each individual and the corresponding Q-table. Set a as the action selected by the individual in the initial state using the ε-greedy method.

Step 2: perform the current action a in the current state s. According to the generation strategy chain $\boldsymbol{GC}$, the number of individuals selecting each generation strategy can be determined. The *mIGD* value can be calculated by Equation (12).

Step 3: the minimum *mIGD* value represents that the individual choosing this generation strategy is in an "excellent" state, and its reward is 1. While the reward of "poor" generation strategy is 1. The new state and the corresponding reward can be obtained.

Step 4: according to $s^{'}$, a new action $a^{'}$ is selected.

Step 5: update the Q-table: $Q(s,a) = Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$.

Step 6: $s = s^{'}$; $a = a^{'}$.

Step 7: update $\boldsymbol{GC}$.

4.3. Overall Implementation of the Proposed Algorithm

The proposed algorithm ACMODE mainly includes two stages, namely initialization and self-adaptation. The pseudocode of our proposed algorithm is described in Algorithm 2. Lines 1 to 2 are the initialization operator. Firstly, the population is randomly generated. Then, the state vector $\boldsymbol{AV}$ and $\boldsymbol{GV}$, the action chain $\boldsymbol{AC}$ and $\boldsymbol{GC}$ and the reward chain $\boldsymbol{RC}$ and $\boldsymbol{GRC}$ are also initialized. Lines 3–9 realize the adaption of CHTs and the adaption of generation strategies. Finally, feasible solutions are selected to enter in the external archive $\boldsymbol{B}$.

Algorithm 2: ACMODE

Input: $G_{\max}$: the maximum number of iterations
Output: final solution set P
Initialize population $P^G = \{x_1^G, \ldots, x_i^G, \ldots x_{NP}^G\}$;
Initialize the external archive $\boldsymbol{B}$, Q-table, the state vector $\boldsymbol{AV}$ and $\boldsymbol{GV}$, the action chain $\boldsymbol{AC}$ and $\boldsymbol{GC}$ and the reward chain $\boldsymbol{RC}$ and $\boldsymbol{GRC}$;
for G=1: G_{max} **do**
    Each individual selects a F value from the set {0.6, 0.8, 1.0};
    Each individual selects a CR value from the set {0.1, 0.2, 1.0};
    Implement the adaptation of generation strategies according to Section 4.2;
    Implement the adaptation of CHTs according to algorithm 1;
    Save the feasible solutions at the first level of non-dominated sorting to $\boldsymbol{B}$;
end for
Output final solution set P according to $\boldsymbol{B}$.

5. Experimental Studies

To evaluate the performance of the ACMODE algorithm, it was compared with four other CMOEAs, which are ACHT-CMODE [7], AGS-CMODE, MOEA/D-CDP [41] and ANSGAIII [42]. In addition, two nonparametric statistical tests, the Wilcoxon rank sum test [43] and the Friedman test [44], are employed to analyze the search performances of all comparison algorithms. "+", "−" and "=", respectively, indicate that the performance of the comparison algorithm is superior, inferior or similar to that of the ACMODE.

5.1. Benchmark Test Functions and Parameter Settings

All experiments are performed on five benchmark test suites, which are CF [45], LIR-CMOP [46], NCTP [47], MW [48] and DAS-CMOP [49]. CF has 10 CMOPs, LIR-CMOP has 14 test functions, NCTP has 18 test functions, MW has 14 test functions and DAS-CMOP has nine test functions. Two comprehensive performance indicators (*IGD* and *HV*) are used to evaluate the algorithm's performance. For all compared algorithms, run times are set to be 30 on each function, the maximum number of iterations is set to be 500, and the population size is set to be 100. Furthermore, the parameter settings of other comparison algorithms are consistent with the original literature.

5.2. Comparison Results

5.2.1. Comparison Results on CF Test Suite

For the CF test suite, Tables 2 and 3 give the comparison results of ACMODE and its comparison algorithms in terms of *IGD* and *HV*, respectively. Note that *IGD* and *HV* can be only used to calculate feasible solutions. The best results are in bold.

As shown in Table 2, the results obtained by Wilcoxon's rank sum test reveal that ACMODE is significantly better than ACHT-CMODE in nine test functions. There is no significant difference between ACMODE and ACHT-CMODE on CF4. In addition, ACMODE performs better than AGS-CMODE in five test functions. ACMODE is similar to AGS-CMODE in five test functions. For the rest of the competitors, MOEA/D-CDP and ANSGAIII are, respectively, worse than ACMODE in nine test functions. ACMODE is similar to MOEA/D-CDP and ANSGAIII in one test function, respectively. The results shown in Table 2 indicate that ACMODE is the best one among all compared algorithms with respect to IGD. The superior performance of the ACMODE is mainly because that appropriate generation strategy and CHT can be adaptively selected in the ACMODE at different stages of evolution.

It can be observed from Table 3 that ACHT-CMODE is outperformed by ACMODE in eight test functions. Moreover, there is no significant difference between ACMODE and ACHT-CMODE on CF9 and CF10. ACMODE performs significantly better than AGS-CMODE in seven test functions. The performance of ACMODE is similar to that of AGS-CMODE on CF2, CF7 and CF10. Furthermore, MOEA/D-CDP and ANSGAIII are significantly surpassed by ACMODE in all test functions. According to the results shown in Table 3, ACMODE is

superior to the other four competitors in terms of HV. The superiority of ACMODE is mainly due to the adaptation of generation strategies and CHTs. The true *PFs* of CF8 and CF10 are disconnected and hindered by several large infeasible regions. This brings challenges to the CMOEAs. MOEA/D-CDP and ANSGAIII are unable to find feasible solutions. However, the experimental results show that the adaptive adjustment of CHTs and generation strategies is beneficial to pass through the infeasible region and then approach the feasible parts.

Table 2. *IGD* results of all comparison algorithms on CF test suite.

	ACHT-CMODE	**AGS-CMODE**	**MOEA/D-CDP**	**ANSGAIII**	**ACMODE**
CF1	5.2867×10^{-2} (7.62×10^{-3}) –	3.7910×10^{-2} (1.19×10^{-2}) –	3.7102×10^{-2} (4.12×10^{-3}) –	3.4372×10^{-2} (3.22×10^{-3}) –	**1.0970×10^{-2} (1.70×10^{-3})**
CF2	9.0126×10^{-2} (3.69×10^{-2}) –	3.0672×10^{-2} (1.15×10^{-2}) =	1.8108×10^{-1} (5.59×10^{-2}) –	8.2723×10^{-2} (4.15×10^{-2}) –	**3.0182×10^{-2} (1.00×10^{-2})**
CF3	2.9612×10^{-1} (9.80×10^{-2}) –	2.6593×10^{-1} (1.10×10^{-1}) –	3.3179×10^{-1} (1.21×10^{-1}) –	2.2112×10^{-1} (5.41×10^{-2}) –	**1.4216×10^{-1} (1.11×10^{-1})**
CF4	7.6876×10^{-2} (1.79×10^{-2}) =	7.3348×10^{-2} (9.28×10^{-3}) =	1.6755×10^{-1} (4.40×10^{-2}) –	1.0219×10^{-1} (3.14×10^{-2}) –	**7.0695×10^{-2} (1.01×10^{-2})**
CF5	3.8964×10^{-1} (1.77×10^{-1}) –	2.5490×10^{-1} (1.02×10^{-1}) =	3.7316×10^{-1} (1.45×10^{-1}) –	2.9797×10^{-1} (1.22×10^{-1}) –	**2.2979×10^{-1} (1.41×10^{-1})**
CF6	6.6338×10^{-2} (2.31×10^{-2}) –	6.1696×10^{-2} (2.76×10^{-2}) –	1.7894×10^{-1} (5.31×10^{-2}) –	7.5225×10^{-2} (2.96×10^{-2}) –	**5.0066×10^{-2} (2.14×10^{-2})**
CF7	3.0948×10^{-1} (1.59×10^{-1}) –	2.5622×10^{-1} (1.86×10^{-1}) =	4.1771×10^{-1} (1.62×10^{-1}) –	3.2974×10^{-1} (1.25×10^{-1}) –	**2.1991×10^{-1} (1.20×10^{-1})**
CF8	4.0369×10^{-1} (1.01×10^{-1}) –	3.2016×10^{-1} (2.46×10^{-2}) –	NaN (NaN)	NaN (NaN)	**3.0085×10^{-1} (3.02×10^{-2})**
CF9	2.2998×10^{-1} (2.75×10^{-2}) –	1.9144×10^{-1} (1.63×10^{-2}) –	**1.6194×10^{-1} (2.33×10^{-2}) =**	1.9518×10^{-1} (1.08×10^{-1}) =	1.6347×10^{-1} (2.48×10^{-2})
CF10	4.8295×10^{0} (4.96×10^{0}) –	6.1970×10^{-1} (8.87×10^{-2}) =	NaN (NaN)	NaN (NaN)	**5.8169×10^{-1} (9.69×10^{-2})**
+/=/−	0/1/9	0/5/5	0/1/9	0/1/9	/

Table 3. *HV* results of all comparison algorithms on CF test suite.

	ACHT-CMODE	**AGS-CMODE**	**MOEA/D-CDP**	**ANSGAIII**	**ACMODE**
CF1	5.0226×10^{-1} (7.44×10^{-3}) –	5.3020×10^{-1} (1.35×10^{-2}) –	5.1961×10^{-1} (4.77×10^{-3}) –	5.2354×10^{-1} (4.12×10^{-3}) –	**5.5277×10^{-1} (2.06×10^{-3})**
CF2	6.1541×10^{-1} (2.87×10^{-2}) –	6.3414×10^{-1} (1.64×10^{-2}) =	5.5158×10^{-1} (3.16×10^{-2}) –	5.9870×10^{-1} (2.31×10^{-2}) –	**6.3741×10^{-1} (1.21×10^{-2})**
CF3	1.4263×10^{-1} (4.49×10^{-2}) –	6.2928×10^{-2} (5.37×10^{-2}) –	1.4502×10^{-1} (3.71×10^{-2}) –	1.6713×10^{-1} (3.70×10^{-2}) –	**2.2932×10^{-1} (6.97×10^{-2})**
CF4	4.1802×10^{-1} (2.82×10^{-2}) –	4.2951×10^{-1} (1.88×10^{-2}) –	3.6115×10^{-1} (3.64×10^{-2}) –	4.1346×10^{-1} (3.10×10^{-2}) –	**4.4357×10^{-1} (1.33×10^{-2})**
CF5	1.7365×10^{-1} (1.16×10^{-1}) –	2.1278×10^{-1} (7.31×10^{-2}) –	2.5767×10^{-1} (7.95×10^{-2}) –	2.6832×10^{-1} (6.40×10^{-2}) –	**3.0478×10^{-1} (8.62×10^{-2})**
CF6	6.3664×10^{-1} (1.30×10^{-2}) –	6.4217×10^{-1} (1.17×10^{-2}) –	5.9050×10^{-1} (3.13×10^{-2}) –	6.3067×10^{-1} (1.78×10^{-2}) –	**6.5454×10^{-1} (1.13×10^{-2})**
CF7	3.0291×10^{-1} (1.75×10^{-1}) –	4.5368×10^{-1} (1.14×10^{-1}) =	3.6811×10^{-1} (1.18×10^{-1}) –	4.2346×10^{-1} (7.56×10^{-2}) –	**4.8571×10^{-1} (8.66×10^{-2})**
CF8	1.6747×10^{-1} (5.29×10^{-2}) –	1.7895×10^{-1} (2.12×10^{-2}) –	NaN (NaN)	NaN (NaN)	**2.1303×10^{-1} (2.79×10^{-2})**
CF9	3.8431×10^{-1} (5.47×10^{-2}) =	3.3650×10^{-1} (3.04×10^{-2}) –	3.6784×10^{-1} (2.87×10^{-2}) –	3.6594×10^{-1} (6.20×10^{-2}) –	**3.9182×10^{-1} (3.27×10^{-2})**
CF10	6.2801×10^{-2} (8.36×10^{-2}) =	9.8047×10^{-2} (1.45×10^{-2}) =	NaN (NaN)	NaN (NaN)	**1.0148×10^{-1} (2.46×10^{-2})**
+/=/−	0/2/8	0/3/7	0/0/10	0/0/10	/

5.2.2. Comparison Results on LIR-CMOP Test Suite

For LIR-CMOP test suite, Tables 4 and 5 list the results of all comparison algorithms in terms of *IGD* and *HV*, respectively. The best results are in bold.

Based on the results obtained by Wilcoxon's rank sum test in Table 4, we can observe that ACMODE significantly outperforms ACHT-CMODE in all 14 test functions. ACMODE performs better than AGS-CMODE in ten test functions and is similar to AGS-CMODE in four test functions. AGS-CMODE carries out the adaptation of generation strategies, and only one constraint handling method (i.e., CDP) is used. However, ACMODE simultaneously realizes the adaptive adjustment of CHTs and generation strategies. Compared with MOEA/D-CDP, ACMODE shows the similar performance on LIR-CMOP4 and LIR-CMOP6. ACMODE is significantly better than MOEA/D-CDP in 12 test functions. However, ACMODE is outperformed by MOEA/D-CDP on LIR-CMOP14. In addition, ANSGAIII is significantly worse than ACMODE in 12 test functions. The performance of the ACMODE is similar to that of ANSGAIII on LIR-CMOP4 and LIR-CMOP6. These experimental results demonstrate that the ACMODE outperforms all the compared algorithms. The effectiveness of the ACMODE can be attributed to the adaptation of CHTs and generation strategies.

When the solutions obtained by the algorithm are all Pareto dominated by the reference points, the *HV* value is zero. The reference points of other comparison algorithms are consistent with the original literature [7,41,42]. From the results regarding *HV* shown in Table 5, ACHT-CMODE performs significantly worse than ACMODE on all 14 test functions. In fact, LIR-CMOP5-LIR-CMOP14 exists within the infeasible barriers when approximating the true *PF*. ACMODE is more likely to choose "DE/rand-to-best/1/bin", which has a better exploration ability. Therefore, ACMODE is able to pass through the infeasible region and find high-quality solutions. ACMODE is superior to AGS-CMODE in 11 test functions, and there is no significant difference between ACMODE and AGS-CMODE on LIR-CMOP4, LIR-CMOP9 and LIR-CMOP12. For the rest of compared algorithms, MOEA/D-CDP and ANSGAIII are, respectively, worse than ACMODE in 12 test functions. The performance of ACMODE is similar to that of MOEA/D-CDP and ANSGAIII in two test functions, respectively. ACMODE particularly stands out among all the comparison algorithms because its CHTs and generation strategies can be automatically selected at different evolutionary stages for different CMOPs.

Table 4. *IGD* results of all comparison algorithms on LIR-CMOP test suite.

	ACHT-CMODE	**AGS-CMODE**	**MOEA/D-CDP**	**ANSGAIII**	**ACMODE**
LIR-CMOP1	3.6782×10^{-1} (3.26×10^{-2}) –	2.3948×10^{-1} (2.93×10^{-2}) –	2.8261×10^{-1} (2.59×10^{-2}) –	3.1440×10^{-1} (3.53×10^{-2}) –	**1.2279×10^{-1} (1.23×10^{-1})**
LIR-CMOP2	3.1322×10^{-1} (4.71×10^{-2}) –	2.1796×10^{-1} (4.85×10^{-2}) –	2.4263×10^{-1} (2.63×10^{-2}) –	2.6801×10^{-1} (2.32×10^{-2}) –	**9.6984×10^{-2} (9.13×10^{-2})**
LIR-CMOP3	3.5187×10^{-1} (3.17×10^{-2}) –	2.8157×10^{-1} (4.36×10^{-2}) –	2.8272×10^{-1} (3.83×10^{-2}) –	3.1967×10^{-1} (3.25×10^{-2}) –	**1.3190×10^{-1} (1.14×10^{-1})**
LIR-CMOP4	3.2254×10^{-1} (7.29×10^{-3}) –	2.7900×10^{-1} (4.83×10^{-2}) =	2.6684×10^{-1} (3.91×10^{-2}) =	2.9089×10^{-1} (2.94×10^{-2}) =	**1.6098×10^{-1} (1.09×10^{-1})**
LIR-CMOP5	1.2205×10^{0} (7.48×10^{-3}) –	1.2156×10^{0} (2.33×10^{-2}) =	1.4539×10^{0} (5.10×10^{-1}) –	1.2484×10^{0} (6.88×10^{-2}) –	**1.1738×10^{0} (1.94×10^{-1})**
LIR-CMOP6	1.3471×10^{0} (1.02×10^{-3}) –	1.2726×10^{0} (2.38×10^{-1}) –	1.4029×10^{0} (2.59×10^{-1}) =	1.3460×10^{0} (3.32×10^{-4}) =	**1.1320×10^{0} (3.99×10^{-1})**
LIR-CMOP7	8.4456×10^{-1} (7.45×10^{-1}) –	1.0911×10^{0} (7.23×10^{-1}) –	1.5268×10^{0} (4.09×10^{-1}) –	1.4345×10^{0} (5.53×10^{-1}) –	**2.3164×10^{-1} (2.95×10^{-1})**
LIR-CMOP8	1.1796×10^{0} (6.73×10^{-1}) –	1.2161×10^{0} (6.40×10^{-1}) –	1.6403×10^{0} (2.10×10^{-1}) –	1.5431×10^{0} (4.10×10^{-1}) –	**8.1410×10^{-1} (7.27×10^{-1})**
LIR-CMOP9	1.0273×10^{0} (7.07×10^{-2}) –	5.5734×10^{-1} (7.24×10^{-2}) =	9.0252×10^{-1} (1.10×10^{-1}) –	1.0191×10^{0} (4.95×10^{-2}) –	**5.3120×10^{-1} (4.73×10^{-2})**

Table 4. *Cont.*

	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
LIR-CMOP10	9.2893×10^{-1} (4.32×10^{-2}) –	4.4541×10^{-1} (8.36×10^{-2}) –	7.9612×10^{-1} (1.46×10^{-1}) –	1.0207×10^{0} (5.34×10^{-2}) –	**3.2702×10^{-1} (7.42×10^{-2})**
LIR-CMOP11	8.9470×10^{-1} (6.88×10^{-2}) –	5.1836×10^{-1} (1.19×10^{-1}) –	8.6793×10^{-1} (8.18×10^{-2}) –	9.2118×10^{-1} (7.37×10^{-2}) –	**3.7547×10^{-1} (1.51×10^{-1})**
LIR-CMOP12	7.2041×10^{-1} (1.19×10^{-1}) –	3.6486×10^{-1} (5.08×10^{-2}) =	6.8840×10^{-1} (1.64×10^{-1}) –	8.6910×10^{-1} (1.61×10^{-1}) –	**3.4246×10^{-1} (5.85×10^{-2})**
LIR-CMOP13	1.3443×10^{0} (5.97×10^{-3}) –	1.3369×10^{0} (3.90×10^{-2}) –	1.3056×10^{0} (4.43×10^{-4}) –	1.3182×10^{0} (4.62×10^{-3}) –	**1.2505×10^{0} (2.54×10^{-1})**
LIR-CMOP14	1.3018×10^{0} (5.25×10^{-3}) –	1.2980×10^{0} (5.31×10^{-3}) –	**1.2618×10^{0} (4.42×10^{-4}) +**	1.2753×10^{0} (3.67×10^{-3}) –	1.2762×10^{0} (3.77×10^{-2})
+/=/−	0/0/14	0/4/10	1/2/11	0/2/12	

Table 5. *HV* results of all comparison algorithms on LIR-CMOP test suite.

	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
LIR-CMOP-1	1.0470×10^{-1} (7.03×10^{-3}) –	1.3078×10^{-1} (1.08×10^{-2}) –	1.1914×10^{-1} (8.23×10^{-3}) –	1.0895×10^{-1} (9.82×10^{-3}) –	**1.9034×10^{-1} (3.67×10^{-2})**
LIR-CMOP-2	1.9590×10^{-1} (2.60×10^{-2}) –	2.4336×10^{-1} (3.18×10^{-2}) –	2.3581×10^{-1} (1.23×10^{-2}) –	2.2184×10^{-1} (1.07×10^{-2}) –	**3.1171×10^{-1} (4.40×10^{-2})**
LIR-CMOP-3	9.2036×10^{-2} (1.01×10^{-2}) –	1.0801×10^{-1} (1.52×10^{-2}) –	1.0552×10^{-1} (1.07×10^{-2}) –	9.8475×10^{-2} (8.47×10^{-3}) –	**1.6380×10^{-1} (3.04×10^{-2})**
LIR-CMOP-4	1.7758×10^{-1} (6.21×10^{-3}) –	1.9572×10^{-1} (2.39×10^{-2}) =	2.0185×10^{-1} (1.59×10^{-2}) =	1.9483×10^{-1} (1.28×10^{-2}) =	**2.5029×10^{-1} (5.06×10^{-2})**
LIR-CMOP-5	0.0000×10^{0} (0.00×10^{0}) –	0.0000×10^{0} (0.00×10^{0}) –	0.0000×10^{0} (0.00×10^{0}) –	0.0000×10^{0} (0.00×10^{0}) –	**2.4662×10^{-1} (9.96×10^{-2})**
LIR-CMOP-6	0.0000×10^{0} (0.00×10^{0}) –	4.2375×10^{-3} (1.63×10^{-2}) –	0.0000×10^{0} (0.00×10^{0}) –	0.0000×10^{0} (0.00×10^{0}) –	**6.3365×10^{-2} (1.73×10^{-2})**
LIR-CMOP-7	1.3136×10^{-1} (1.17×10^{-1}) –	9.0241×10^{-2} (1.18×10^{-1}) –	2.2081×10^{-2} (6.09×10^{-2}) –	4.5117×10^{-2} (9.18×10^{-2}) –	**2.3786×10^{-1} (2.69×10^{-2})**
LIR-CMOP-8	7.4568×10^{-2} (1.01×10^{-1}) –	6.9517×10^{-2} (9.96×10^{-2}) –	5.1055×10^{-3} (2.80×10^{-2}) –	1.9617×10^{-2} (5.99×10^{-2}) –	**2.2895×10^{-1} (3.90×10^{-2})**
LIR-CMOP-9	1.0130×10^{-1} (3.17×10^{-2}) –	3.1252×10^{-1} (4.57×10^{-2}) =	1.5118×10^{-1} (5.40×10^{-2}) –	1.0356×10^{-1} (2.38×10^{-2}) –	**3.3605×10^{-1} (3.16×10^{-2})**
LIR-CMOP-10	7.7537×10^{-2} (2.52×10^{-2}) –	4.5558×10^{-1} (5.70×10^{-2}) –	1.4128×10^{-1} (7.65×10^{-2}) –	5.6946×10^{-2} (1.03×10^{-2}) –	**5.2440×10^{-1} (4.21×10^{-2})**
LIR-CMOP-11	1.7900×10^{-1} (1.49×10^{-2}) –	3.4756×10^{-1} (1.07×10^{-1}) –	2.0844×10^{-1} (5.00×10^{-2}) –	1.7725×10^{-1} (2.22×10^{-2}) –	**4.4958×10^{-1} (1.20×10^{-1})**
LIR-CMOP-12	3.1138×10^{-1} (5.58×10^{-2}) –	4.2061×10^{-1} (4.38×10^{-2}) =	3.2802×10^{-1} (7.24×10^{-2}) –	2.4222×10^{-1} (8.08×10^{-2}) –	**4.3065×10^{-1} (4.98×10^{-2})**
LIR-CMOP-13	4.6953×10^{-5} (1.00×10^{-4}) –	6.8231×10^{-4} (3.51×10^{-3}) –	4.2034×10^{-4} (3.94×10^{-5}) =	2.0855×10^{-4} (1.75×10^{-4}) –	**3.0773×10^{-2} (8.88×10^{-2})**
LIR-CMOP-14	1.7391×10^{-4} (2.35×10^{-4}) –	2.8551×10^{-4} (2.64×10^{-4}) –	9.5720×10^{-4} (4.84×10^{-5}) –	6.3627×10^{-4} (3.57×10^{-4}) =	**2.2823×10^{-3} (8.72×10^{-3})**
+/=/−	0/0/14	0/3/11	0/2/12	0/2/12	

5.2.3. Comparison Results on NCTP Test Suite

For the NCTP test suite, Tables 6 and 7 give the comparison results of ACMODE and its comparison algorithms in terms of *IGD* and *HV*, respectively. The best results are in bold.

According to the results obtained by Wilcoxon's rank sum test in Table 6, ACHT-CMODE is better than the proposed algorithm ACMODE on NCTP2, NCTP4 and NCTP5. However, it is inferior to ACMODE in 14 test functions. ACMODE is similar to ACHT-CMODE on NCTP1. The main reason is that ACHT-CMODE realizes the adaptation selection of CHTs,

and its search strategies are generally constant during the entire evolutionary process. In addition, ACMODE performs better than AGS-CMODE in 12 test functions. AGS-CMODE is better than the proposed algorithm on NCTP1, NCTP2, NCTP13 and NCTP14. There is no significant difference between AGS-CMODE and ACMODE on NCTP4 and NCTP5. For the rest of the competitors, MOEA/D-CDP performs worse than ACMODE in 16 test functions. MOEA/D-CDP and ACMODE have similar performance on NCTP1 and NCTP2. Additionally, ANSGAIII is significantly surpassed by ACMODE in all test functions. The results shown in Table 6 indicate that ACMODE is the best one among all compared algorithms. The reason may be that appropriate generation strategies and CHTs can be automatically selected at different stages of evolution for different types of CMOPs.

The results shown in Table 7 reveal that ACMODE performs better than ACHT-CMODE in 12 test functions, but ACMODE is surpassed by ACHT-CMODE on NCTP7, NCTP8 and NCTP13. ACMODE is superior to AGS-CMODE in eight test functions. However, AGS-CMODE outperforms ACMODE on NCTP7, NCTP8 and NCTP13. There is no significant difference between AGS-CMODE and ACMODE in seven test functions. The results demonstrate that both the adaptive CHT and the adaptation of generation strategy play an important role in improving the performance of CMOEAs. For the rest of compared algorithms, ACMODE is surpassed by MOEA/D-CDP on NCTP7 and NCTP8. MOEA/D-CDP performs worse than ACMODE in eight test functions. ACMODE and MOEA/D-CDP have no significant difference in eight test functions. Furthermore, ACMODE performs better than ANSGAIII in 11 test functions. ACMODE is significantly outperformed by ANSGAIII on NCTP3, NCTP5 and NCTP6. In addition, there is no significant difference between ACMODE and ANSGAIII in four test functions. Overall, ACMODE is capable of providing better feasible *PF* regarding convergence and diversity in most test functions.

Table 6. *IGD* results of all comparison algorithms on NCTP test suite.

	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
NCTP1	1.3860×10^{-1} (3.74×10^{-2}) =	**1.0771×10^{-1} (9.01×10^{-3}) +**	2.4462×10^{-1} (1.78×10^{-1}) =	NaN (NaN)	1.5979×10^{-1} (6.96×10^{-2})
NCTP2	**2.1228×10^{-1} (4.01×10^{-2}) +**	2.3431×10^{-1} (1.85×10^{-2}) +	3.3089×10^{-1} (9.41×10^{-2}) =	NaN (NaN)	2.5951×10^{-1} (3.15×10^{-2})
NCTP3	1.0298×10^{-1} (6.58×10^{-2}) –	1.0074×10^{-1} (1.87×10^{-2}) –	1.7656×10^{-1} (1.48×10^{-1}) –	5.9349×10^{-1} (8.93×10^{-1}) –	**7.4401×10^{-2} (1.62×10^{-2})**
NCTP4	1.3090×10^{-1} (1.83×10^{-2}) +	1.0642×10^{-1} (7.71×10^{-3}) =	4.3209×10^{-1} (8.37×10^{-1}) –	4.6246×10^{-1} (7.31×10^{-1}) –	**1.0313×10^{-1} (8.84×10^{-3})**
NCTP5	**2.1392×10^{-1} (4.81×10^{-2}) +**	2.3318×10^{-1} (1.69×10^{-2}) =	5.0585×10^{-1} (6.93×10^{-1}) –	7.6896×10^{-1} (9.25×10^{-1}) –	2.3259×10^{-1} (2.19×10^{-2})
NCTP6	1.0380×10^{-1} (4.20×10^{-2}) –	9.2220×10^{-2} (1.78×10^{-2}) –	5.7480×10^{-1} (1.01×10^{0}) –	7.5063×10^{-1} (8.24×10^{-1}) –	**7.2113×10^{-2} (1.91×10^{-2})**
NCTP7	2.7279×10^{-1} (2.99×10^{-1}) –	8.2332×10^{-2} (2.11×10^{-2}) –	5.4114×10^{-1} (4.56×10^{-1}) –	NaN (NaN)	**7.6018×10^{-2} (2.79×10^{-2})**
NCTP8	2.0817×10^{-1} (1.04×10^{-1}) –	8.0694×10^{-2} (2.61×10^{-2}) –	6.6398×10^{-1} (5.79×10^{-1}) –	NaN (NaN)	**6.9113×10^{-2} (2.51×10^{-2})**
NCTP9	1.1628×10^{-1} (5.60×10^{-2}) –	1.0837×10^{-1} (3.46×10^{-2}) –	8.1585×10^{-1} (1.13×10^{0}) –	4.4256×10^{-1} (5.35×10^{-1}) –	**7.0539×10^{-2} (3.59×10^{-2})**
NCTP10	1.4095×10^{-1} (1.59×10^{-1}) –	7.7485×10^{-2} (1.90×10^{-2}) –	5.3551×10^{-1} (1.10×10^{0}) –	8.6430×10^{-1} (1.08×10^{0}) –	**4.1228×10^{-2} (5.25×10^{-3})**
NCTP11	2.2205×10^{-1} (2.11×10^{-1}) –	6.9075×10^{-2} (1.39×10^{-2}) –	6.7656×10^{-1} (5.20×10^{-1}) –	8.6755×10^{-1} (1.41×10^{0}) –	**4.9439×10^{-2} (1.17×10^{-2})**
NCTP12	1.6543×10^{-1} (1.78×10^{-1}) –	1.1236×10^{-1} (2.77×10^{-2}) –	5.8721×10^{-1} (8.06×10^{-1}) –	8.9402×10^{-1} (1.40×10^{0}) –	**4.9613×10^{-2} (8.68×10^{-3})**
NCTP13	1.9554×10^{-1} (3.87×10^{-1}) –	**4.4096×10^{-2} (4.56×10^{-3}) +**	NaN (NaN)	5.3113×10^{-1} (5.81×10^{-1}) –	5.4671×10^{-2} (2.72×10^{-2})

Table 6. *Cont.*

	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
NCTP14	2.2531×10^{-1} (3.94×10^{-1}) −	**4.4640×10^{-2} (8.54×10^{-3}) +**	5.9747×10^{-1} (6.17×10^{-1}) −	NaN (NaN)	6.1650×10^{-2} (1.40×10^{-2})
NCTP15	1.5001×10^{-1} (2.02×10^{-1}) −	4.2951×10^{-2} (4.80×10^{-3}) −	4.5939×10^{-1} (5.08×10^{-1}) −	NaN (NaN)	**3.8772×10^{-2} (3.05×10^{-3})**
NCTP16	1.3413×10^{-1} (1.77×10^{-1}) −	4.4146×10^{-2} (4.35×10^{-3}) −	5.0853×10^{-1} (7.36×10^{-1}) −	4.9278×10^{-1} (6.87×10^{-1}) −	**3.6088×10^{-2} (2.31×10^{-3})**
NCTP17	1.0507×10^{-1} (1.01×10^{-1}) −	4.4572×10^{-2} (6.69×10^{-3}) −	3.8320×10^{-1} (4.51×10^{-1}) −	4.8897×10^{-1} (5.88×10^{-1}) −	**3.6699×10^{-2} (2.63×10^{-3})**
NCTP18	$1.410^{0} \times 10^{-1}$ (1.82×10^{-1}) −	4.4446×10^{-2} (6.20×10^{-3}) −	6.8239×10^{-1} (1.11×10^{0}) −	6.9280×10^{-1} (9.53×10^{-1}) −	**3.6178×10^{-2} (2.93×10^{-3})**
+/=/−	3/1/14	4/2/12	0/2/16	0/0/18	/

Table 7. *HV* results of all comparison algorithms on NCTP test suite.

	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
NCTP1	5.4088×10^{-1} (2.76×10^{-2}) −	5.6237×10^{-1} (8.20×10^{-3}) −	**6.0289×10^{-1} (6.26×10^{-2}) =**	NaN (NaN)	5.7718×10^{-1} (1.79×10^{-2})
NCTP2	5.4146×10^{-1} (1.98×10^{-2}) −	5.6460×10^{-1} (1.25×10^{-2}) =	**5.7051×10^{-1} (4.09×10^{-2}) =**	NaN (NaN)	5.5957×10^{-1} (9.94×10^{-3})
NCTP3	5.5944×10^{-1} (3.51×10^{-2}) −	6.0019×10^{-1} (1.14×10^{-2}) =	**6.2467×10^{-1} (5.64×10^{-2}) =**	6.2739×10^{-1} (1.83×10^{-1}) +	5.9190×10^{-1} (1.09×10^{-2})
NCTP4	5.4199×10^{-1} (1.96×10^{-2}) −	5.6015×10^{-1} (8.50×10^{-3}) =	5.5349×10^{-1} (1.09×10^{-1}) =	5.4983×10^{-1} (1.94×10^{-1}) −	**5.5990×10^{-1} (8.87×10^{-3})**
NCTP5	5.3512×10^{-1} (2.34×10^{-2}) −	5.6368×10^{-1} (1.12×10^{-2}) =	5.4810×10^{-1} (1.09×10^{-1}) =	**5.7141×10^{-1} (2.15×10^{-1}) +**	5.5636×10^{-1} (1.23×10^{-2})
NCTP6	5.6637×10^{-1} (3.18×10^{-2}) −	5.9506×10^{-1} (1.10×10^{-2}) =	5.5482×10^{-1} (1.92×10^{-1}) −	**6.5762×10^{-1} (2.03×10^{-1}) +**	5.9033×10^{-1} (9.67×10^{-3})
NCTP7	6.2801×10^{-1} (6.24×10^{-2}) +	6.4922×10^{-1} (2.70×10^{-3}) +	**6.5732×10^{-1} (1.58×10^{-1}) +**	NaN (NaN)	5.1276×10^{-1} (2.71×10^{-1})
NCTP8	6.1795×10^{-1} (1.88×10^{-2}) +	**6.2606×10^{-1} (4.21×10^{-3}) +**	5.7234×10^{-1} (1.54×10^{-1}) +	NaN (NaN)	5.3824×10^{-1} (2.03×10^{-1})
NCTP9	6.0831×10^{-1} (4.51×10^{-3}) =	6.0372×10^{-1} (4.75×10^{-3}) −	6.0243×10^{-1} (1.86×10^{-1}) =	5.7150×10^{-1} (1.80×10^{-1}) =	**6.1349×10^{-1} (1.70×10^{-1})**
NCTP10	6.4548×10^{-1} (2.30×10^{-2}) −	6.4932×10^{-1} (2.52×10^{-3}) −	6.0053×10^{-1} (1.77×10^{-1}) =	4.8179×10^{-1} (2.16×10^{-1}) =	**6.5318×10^{-1} (1.64×10^{-3})**
NCTP11	6.1443×10^{-1} (4.21×10^{-2}) =	6.2653×10^{-1} (3.53×10^{-3}) −	5.6357×10^{-1} (1.67×10^{-1}) =	5.3849×10^{-1} (1.98×10^{-1}) =	**6.2895×10^{-1} (5.15×10^{-3})**
NCTP12	6.4981×10^{-1} (3.01×10^{-2}) −	6.5731×10^{-1} (3.24×10^{-3}) −	6.2751×10^{-1} (1.84×10^{-1}) −	5.2498×10^{-1} (2.49×10^{-1}) =	**6.6473×10^{-1} (1.99×10^{-3})**
NCTP13	6.6543×10^{-1} (1.38×10^{-1}) +	**7.0834×10^{-1} (8.72×10^{-4}) +**	NaN (NaN)	5.2705×10^{-1} (2.16×10^{-1}) −	6.1506×10^{-1} (2.15×10^{-1})
NCTP14	5.7668×10^{-1} (1.32×10^{-1}) −	5.0439×10^{-1} (2.72×10^{-1}) −	NaN (NaN)	NaN (NaN)	**6.4519×10^{-1} (1.45×10^{-3})**
NCTP15	4.0668×10^{-1} (5.02×10^{-2}) −	4.3265×10^{-1} (5.45×10^{-4}) −	3.2546×10^{-1} (1.16×10^{-1}) −	NaN (NaN)	**4.3636×10^{-1} (1.26×10^{-3})**
NCTP16	6.8158×10^{-1} (6.86×10^{-2}) =	7.0841×10^{-1} (9.34×10^{-4}) =	5.8498×10^{-1} (1.89×10^{-1}) −	5.6543×10^{-1} (2.23×10^{-1}) −	**7.0887×10^{-1} (7.29×10^{-4})**
NCTP17	6.1449×10^{-1} (4.32×10^{-2}) −	5.5516×10^{-1} (1.41×10^{-3}) −	5.2751×10^{-1} (1.49×10^{-1}) −	4.8065×10^{-1} (2.05×10^{-1}) −	**6.4505×10^{-1} (7.83×10^{-4})**
NCTP18	4.0931×10^{-1} (4.69×10^{-2}) −	4.3727×10^{-1} (8.19×10^{-4}) =	3.2606×10^{-1} (1.46×10^{-1}) −	3.0694×10^{-1} (1.43×10^{-1}) −	**4.3741×10^{-1} (5.31×10^{-4})**
+/=/−	3/3/12	3/7/8	2/8/8	$\frac{3}{4}$/11	/

5.2.4. Comparison Results on MW Test Suite

For the MW test suite, Tables 8 and 9 give the comparison results of ACMODE and its comparison algorithms in terms of *IGD* and *HV*, respectively. The best results are in bold.

As can be shown in Table 8, the performance of ACHT-CMODE is better than that of ACMODE on MW4 and MW14. However, it is inferior to ACMODE in 11 test functions. The performance of ACMODE is similar to that of ACHT-CMODE on MW9. In addition, ACMODE performs better than AGS-CMODE in 12 test functions. There is no significant difference between AGS-CMODE and ACMODE on MW4 and MW9. ACHT-CMODE realizes the adaption of CHTs, and AGS-CMODE realizes the adaption of generation strategies. However, the performance of ACMODE is better than that of these two algorithms. Its performance is improved by the adaptive adjustment of CHTs and generation strategies. For the rest of the competitors, MOEA/D-CDP performs worse than ACMODE in eight test functions. MOEA/D-CDP and ACMODE have similar performance on MW8 and MW12. MOEA/D-CDP outperforms ACMODE in four test functions. Additionally, ANSGAIII is significantly surpassed by ACMODE in 12 test functions. ANSGAIII performs better than ACMODE on MW13 and MW14. The results shown in Table 8 indicate that the performance of ACMODE is the best among all compared algorithms.

It can be observed from Table 9 that ACHT-CMODE is surpassed by ACMODE in eight test functions. There is no significant difference between ACHT-CMODE and ACMODE in six test functions. ACMODE is superior to AGS-CMODE in nine test functions. ACMODE and AGS-CMODE have no significant difference in five test functions. Table 9 also shows that ACMODE is surpassed by MOEA/D-CDP on MW4 and MW13. MOEA/D-CDP performs worse than ACMODE in six test functions. There is no significant difference between MOEA/D-CDP and ACMODE in six test functions. Furthermore, ACMODE performs better than ANSGAIII in 12 test functions and is significantly outperformed by ANSGAIII on MW14. ACMODE and ANSGAIII have similar performances on MW13. MOEA/D-CDP and ANSGAIII are unable to find the feasible solutions on MW1. This is because MW1 has a disconnected constrained PF with narrow feasible regions. However, the proposed algorithm can automatically select suitable CHT and generation strategies to solve different types of CMOPs.

Table 8. *IGD* results of all comparison algorithms on MW test suite.

	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
MW1	3.5685×10^{-2} (1.25×10^{-1}) –	2.8590×10^{-2} (1.26×10^{-1}) –	NaN (NaN)	NaN (NaN)	**3.6886×10^{-3} (3.42×10^{-4})**
MW2	1.2796×10^{-1} (5.40×10^{-2}) –	9.4127×10^{-2} (4.26×10^{-2}) –	2.0047×10^{-2} (7.02×10^{-3}) –	2.5678×10^{-2} (2.12×10^{-2}) –	**6.0942×10^{-3} (3.31×10^{-4})**
MW3	1.1164×10^{-2} (8.37×10^{-4}) –	6.0933×10^{-2} (1.86×10^{-1}) –	1.0535×10^{-2} (1.73×10^{-2}) –	1.6396×10^{-2} (2.55×10^{-2}) –	**7.2653×10^{-3} (7.04×10^{-4})**
MW4	6.1867×10^{-2} (3.35×10^{-3}) +	8.4142×10^{-2} (3.31×10^{-3}) =	**4.9103×10^{-2} (3.18×10^{-2}) +**	NaN (NaN)	7.9096×10^{-2} (5.38×10^{-2})
MW5	1.0475×10^{-1} (2.51×10^{-1}) –	1.7277×10^{-1} (3.14×10^{-1}) –	1.2854×10^{-1} (2.71×10^{-1}) –	NaN (NaN)	**1.9151×10^{-2} (6.58×10^{-2})**
MW6	4.9754×10^{-1} (2.12×10^{-1}) –	3.8194×10^{-1} (2.44×10^{-1}) –	1.6465×10^{-2} (7.24×10^{-3}) –	8.3408×10^{-2} (1.46×10^{-1}) –	**5.3924×10^{-3} (8.97×10^{-4})**
MW7	1.5003×10^{-2} (1.55×10^{-3}) –	1.5839×10^{-2} (1.84×10^{-3}) –	**5.2247×10^{-3} (2.17×10^{-4}) +**	7.0319×10^{-2} (1.47×10^{-1}) –	9.0254×10^{-3} (7.36×10^{-4})
MW8	3.9036×10^{-1} (3.97×10^{-1}) –	2.5627×10^{-1} (2.58×10^{-1}) –	**5.2612×10^{-2} (3.02×10^{-3}) =**	1.1601×10^{-1} (1.59×10^{-1}) –	6.4412×10^{-2} (3.61×10^{-3})
MW9	**1.8159×10^{-2} (4.22×10^{-3}) =**	2.6656×10^{-2} (4.22×10^{-3}) =	4.3805×10^{-2} (1.16×10^{-1}) –	1.4554×10^{-1} (2.31×10^{-1}) –	2.0545×10^{-2} (4.97×10^{-3})
MW10	5.4517×10^{-1} (3.49×10^{-1}) –	4.7261×10^{-1} (1.24×10^{-1}) –	5.7076×10^{-2} (9.91×10^{-2}) –	NaN (NaN)	**5.0451×10^{-2} (1.40×10^{-1})**
MW11	3.1015×10^{-2} (3.07×10^{-2}) –	1.7603×10^{-2} (2.47×10^{-3}) –	3.1642×10^{-1} (3.49×10^{-1}) –	6.5410×10^{-1} (1.92×10^{-1}) –	**1.1193×10^{-2} (1.88×10^{-3})**

Table 8. *Cont.*

	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
MW12	1.4828×10^{-2} (2.13×10^{-3}) –	1.5529×10^{-2} (2.29×10^{-3}) –	9.1041×10^{-3} (9.61×10^{-3}) =	2.4633×10^{-1} (2.16×10^{-1}) –	**8.1329×10^{-3} (1.31×10^{-3})**
MW13	4.2169×10^{-1} (6.07×10^{-1}) –	1.1238×10^{0} (1.14×10^{0}) –	**1.0992×10^{-1} (9.78×10^{-2}) +**	2.8903×10^{-1} (2.28×10^{-1}) +	5.4055×10^{-1} (5.71×10^{-1})
MW14	2.0062×10^{-1} (5.33×10^{-2}) +	5.9693×10^{-1} (8.59×10^{-1}) –	2.0936×10^{-1} (3.13×10^{-3}) +	**1.4393×10^{-1} (6.22×10^{-2}) +**	2.9577×10^{-1} (5.87×10^{-2})
+/=/−	2/1/11	0/2/12	4/2/8	2/0/12	/

Table 9. *HV* results of all comparison algorithms on MW test suite.

	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
MW1	3.8045×10^{-1} (2.03×10^{-1}) –	4.7016×10^{-1} (9.04×10^{-2}) =	NaN (NaN)	NaN (NaN)	**4.8900×10^{-1} (2.05×10^{-4})**
MW2	3.9179×10^{-1} (8.73×10^{-2}) –	4.5204×10^{-1} (5.21×10^{-2}) –	5.5505×10^{-1} (1.15×10^{-2}) –	5.4544×10^{-1} (2.97×10^{-2}) –	**5.7959×10^{-1} (4.00×10^{-4})**
MW3	5.3649×10^{-1} (1.47×10^{-3}) –	5.0459×10^{-1} (1.12×10^{-1}) –	5.3963×10^{-1} (1.30×10^{-2}) –	5.3463×10^{-1} (1.94×10^{-2}) –	**5.4508×10^{-1} (8.20×10^{-2})**
MW4	8.1170×10^{-1} (5.91×10^{-3}) =	7.7877×10^{-1} (6.46×10^{-3}) –	**8.3018×10^{-1} (4.80×10^{-2}) +**	NaN (NaN)	7.9922×10^{-1} (2.29×10^{-2})
MW5	2.8276×10^{-1} (7.62×10^{-2}) –	2.6895×10^{-1} (9.99×10^{-2}) –	2.9676×10^{-1} (7.04×10^{-2}) =	NaN (NaN)	**3.1409×10^{-1} (3.17×10^{-2})**
MW6	1.2894×10^{-1} (7.70×10^{-2}) –	1.4219×10^{-1} (7.45×10^{-2}) –	3.0778×10^{-1} (1.06×10^{-2}) =	2.8431×10^{-1} (4.58×10^{-2}) –	**3.2721×10^{-1} (1.69×10^{-3})**
MW7	4.0222×10^{-1} (2.04×10^{-3}) =	4.0040×10^{-1} (1.82×10^{-3}) =	**4.1054×10^{-1} (3.39×10^{-4}) =**	3.8371×10^{-1} (5.71×10^{-2}) –	4.0999×10^{-1} (2.99×10^{-2})
MW8	5.1365×10^{-1} (5.89×10^{-3}) –	3.1684×10^{-1} (1.09×10^{-1}) –	5.3025×10^{-1} (1.35×10^{-2}) =	5.0202×10^{-1} (5.97×10^{-2}) –	**5.4384×10^{-1} (1.89×10^{-1})**
MW9	**3.7959×10^{-1} (7.01×10^{-3}) =**	3.6561×10^{-1} (5.48×10^{-3}) =	3.5258×10^{-1} (8.63×10^{-2}) –	3.5021×10^{-1} (4.71×10^{-2}) –	3.7506×10^{-1} (6.58×10^{-3})
MW10	4.2631×10^{-1} (7.77×10^{-2}) =	1.7716×10^{-1} (5.88×10^{-2}) –	4.0371×10^{-1} (5.71×10^{-2}) –	NaN (NaN)	**4.4409×10^{-1} (6.81×10^{-3})**
MW11	4.3838×10^{-1} (1.47×10^{-2}) –	4.4266×10^{-1} (1.17×10^{-3}) =	3.7572×10^{-1} (8.38×10^{-2}) –	2.8696×10^{-1} (4.44×10^{-2}) –	**4.4538×10^{-1} (1.10×10^{-3})**
MW12	5.9577×10^{-1} (1.37×10^{-3}) =	5.9427×10^{-1} (2.44×10^{-3}) =	6.0002×10^{-1} (7.64×10^{-3}) =	4.3413×10^{-1} (1.20×10^{-1}) –	**6.0154×10^{-1} (1.09×10^{-3})**
MW13	3.8252×10^{-1} (1.42×10^{-1}) =	1.9149×10^{-1} (1.67×10^{-1}) –	**4.3410×10^{-1} (3.39×10^{-2}) +**	3.8801×10^{-1} (4.65×10^{-2}) =	3.8573×10^{-1} (1.66×10^{-1})
MW14	4.2247×10^{-1} (3.60×10^{-2}) –	3.2069×10^{-1} (1.08×10^{-1}) –	4.3603×10^{-1} (3.70×10^{-3}) =	**4.5033×10^{-1} (2.99×10^{-2}) +**	4.4174×10^{-1} (1.00×10^{-1})
+/=/−	0/6/8	0/5/9	2/6/6	1/1/12	/

5.2.5. Comparison Results on DAS-CMOP Test Suite

For DAS-CMOP test suite, Tables 10 and 11 give the comparison results of ACMODE and its comparison algorithms in terms of *IGD* and *HV*, respectively. The best results are in bold.

It can be observed from Table 10 that ACHT-CMODE performs better than ACMODE on DAS-CMOP7, but it is inferior to ACMODE in eight test functions. ACMODE performs better than AGS-CMODE in eight test functions. There is no significant difference between AGS-CMODE and ACMODE on DAS-CMOP9. ACMODE is significantly better than MOEA/D-CDP in five test functions. However, ACMODE is outperformed by MOEA/D-CDP in four test functions. In addition, ANSGAIII is significantly worse than ACMODE in six test functions. ANSGAIII performs better than ACMODE in three test functions. These experimental results demonstrate that the ACMODE outperforms all the compared algorithms.

The results shown in Table 11 reveal that ACMODE performs better than ACHT-CMODE in six test functions, but ACMODE is surpassed by ACHT-CMODE on DAS-CMOP7. ACMODE and ACHT-CMODE have no significant difference on DAS-CMOP3 and DAS-CMOP5. ACMODE is superior to AGS-CMODE in seven test functions. There is no significant difference between AGS-CMODE and ACMODE on DAS-CMOP3 and DAS-CMOP9. Table 11 also indicates that ACMODE is surpassed by MOEA/D-CDP on DAS-CMOP3, DAS-CMOP4, and DAS-CMOP7. MOEA/D-CDP performs worse than ACMODE in five test functions. Furthermore, ACMODE performs better than ANSGAIII in six test functions and is significantly outperformed by ANSGAIII on DAS-CMOP3, DAS-CMOP7 and DAS-CMOP8.

Table 10. *IGD* results of all comparison algorithms on DAS-CMOP test suite.

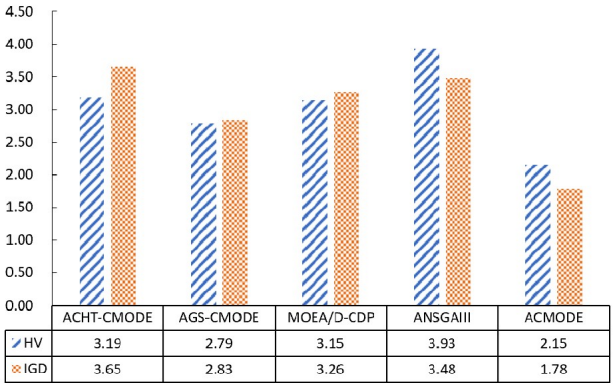
	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
DAS-CMOP1	6.8402×10^{-1} (3.95×10^{-2}) −	3.0521×10^{-1} (3.01×10^{-1}) −	7.0233×10^{-1} (2.86×10^{-2}) −	7.2659×10^{-1} (3.20×10^{-2}) −	**9.7386×10^{-2} (2.12×10^{-1})**
DAS-CMOP2	2.0314×10^{-1} (2.47×10^{-2}) −	1.4664×10^{-1} (7.51×10^{-2}) −	2.0182×10^{-1} (2.51×10^{-2})−	2.3340×10^{-1} (2.52×10^{-2}) −	**5.6632×10^{-2} (6.61×10^{-2})**
DAS-CMOP3	4.5027×10^{-1} (1.94×10^{-1}) −	4.7333×10^{-1} (2.50×10^{-1}) −	**3.4909×10^{-1} (2.85×10^{-2}) +**	3.6324×10^{-1} (7.28×10^{-2}) +	3.9314×10^{-1} (2.16×10^{-1})
DAS-CMOP4	5.8602×10^{-1} (2.50×10^{-1}) −	NaN (NaN)	**3.7875×10^{-1} (1.59×10^{-1}) +**	5.2573×10^{-1} (7.70×10^{-2}) −	4.0609×10^{-1} (3.22×10^{-1})
DAS-CMOP5	5.4619×10^{-1} (2.66×10^{-1}) −	NaN (NaN)	NaN (NaN)	NaN (NaN)	**5.1266×10^{-1} (1.56×10^{-2})**
DAS-CMOP6	5.7986×10^{-1} (2.70×10^{-1}) −	NaN (NaN)	NaN (NaN)	NaN (NaN)	**5.4968×10^{-1} (2.89×10^{-1})**
DAS-CMOP7	4.3361×10^{-1} (3.71×10^{-1}) +	NaN (NaN)	**7.5066×10^{-2} (3.53×10^{-2}) +**	8.0605×10^{-2} (4.30×10^{-2}) +	5.0813×10^{-1} (4.22×10^{-1})
DAS-CMOP8	4.2182×10^{-1} (3.30×10^{-1}) −	1.3639×10^{0} (1.94×10^{-1}) −	2.1033×10^{-1} (2.39×10^{-1}) +	**1.3997×10^{-1} (1.25×10^{-1}) +**	2.8305×10^{-1} (2.29×10^{-1})
DAS-CMOP9	2.8285×10^{-1} (5.11×10^{-2}) −	**1.7237×10^{-1} (1.20×10^{-1}) =**	3.0124×10^{-1} (9.30×10^{-2}) −	3.4032×10^{-1} (4.87×10^{-2}) −	2.0376×10^{-1} (1.30×10^{-1})
+/=/−	1/0/8	0/1/8	4/0/5	3/0/6	/

Table 11. *HV* results of all comparison algorithms on DAS-CMOP test suite.

	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
DAS-CMOP1	1.1540×10^{-2} (8.16×10^{-3}) −	1.2188×10^{-1} (9.03×10^{-2}) −	7.7281×10^{-3} (5.66×10^{-3}) −	5.1066×10^{-3} (6.26×10^{-3}) −	**1.8329×10^{-1} (6.23×10^{-2})**
DAS-CMOP2	2.5058×10^{-1} (6.81×10^{-3}) −	2.8191×10^{-1} (3.16×10^{-2}) −	2.5360×10^{-1} (9.27×10^{-3}) −	2.4274×10^{-1} (5.00×10^{-3}) −	**3.2420×10^{-1} (3.47×10^{-2})**
DAS-CMOP3	1.6687×10^{-1} (8.48×10^{-2}) =	1.5288×10^{-1} (1.05×10^{-1}) =	**2.0953×10^{-1} (6.74×10^{-3}) +**	2.0341×10^{-1} (3.40×10^{-2}) +	1.8499×10^{-1} (8.86×10^{-2})
DAS-CMOP4	3.4570×10^{-2} (3.94×10^{-2}) −	NaN (NaN)	**8.0763×10^{-2} (5.22×10^{-2}) +**	3.4611×10^{-2} (1.29×10^{-2}) −	7.3804×10^{-2} (8.53×10^{-2})
DAS-CMOP5	**1.0397×10^{-1} (9.64×10^{-2}) =**	NaN (NaN)	NaN (NaN)	NaN (NaN)	8.4077×10^{-2} (4.31×10^{-3})
DAS-CMOP6	9.3738×10^{-2} (1.02×10^{-1}) −	NaN (NaN)	NaN (NaN)	NaN (NaN)	**1.3128×10^{-1} (1.18×10^{-1})**
DAS-CMOP7	1.4421×10^{-1} (1.05×10^{-1}) +	NaN (NaN)	**2.6506×10^{-1} (2.56×10^{-2}) +**	2.4506×10^{-1} (1.52×10^{-2}) +	1.2062×10^{-1} (1.14×10^{-1})
DAS-CMOP8	8.6964×10^{-2} (7.86×10^{-2}) −	0.0000×10^{0} (0.00×10^{0}) −	1.5944×10^{-1} (5.54×10^{-2}) =	**1.7561×10^{-1} (2.69×10^{-2}) +**	1.1666×10^{-1} (8.19×10^{-2})
DAS-CMOP9	1.2571×10^{-1} (1.02×10^{-2}) −	**1.6214×10^{-1} (3.63×10^{-2}) =**	1.0977×10^{-1} (2.84×10^{-2}) −	1.0218×10^{-1} (1.54×10^{-2}) −	1.5383×10^{-1} (3.95×10^{-2})
+/=/−	1/2/6	0/2/7	3/1/5	3/0/6	/

5.2.6. Overall Comparison Results on All Test Suites

In this section, the overall performance of ACMODE on all 65 test functions is evaluated by the Friedman test [44]. The average rankings of *HV* and *IGD* of all comparison algorithms are shown in Figure 1. A smaller average ranking value denotes a better performance. For the *IGD* values, the experimental results demonstrate that the proposed algorithm ranks first among all the comparison algorithms. The overall performance of ACMODE is the best in terms of *HV*. It can be concluded that the overall performance of ACMODE is better than that of the other four algorithms. This is because ACMODE can choose the CHT and generation strategy with the best performance during different stages of the search process.



	ACHT-CMODE	AGS-CMODE	MOEA/D-CDP	ANSGAIII	ACMODE
HV	3.19	2.79	3.15	3.93	2.15
IGD	3.65	2.83	3.26	3.48	1.78

Figure 1. Performance rankings of all comparison algorithms.

5.3. Experimental Analysis

5.3.1. The Effectiveness of Adaptive Constraint Handling Technology

In ACMODE, three widely used CHTs are selected to realize the adaptation of CHTs. In order to further verify the effectiveness of the proposed adaptive constraint handling technology, the evolution curves of CHTs in two test functions (CF6 and NCTP11) are presented in this experiment.

Figure 2 depicts the number of individuals of each CHT during the entire evolutionary process. In this section, three variants are investigated by adopting different CHTs. To be specific, only "ATM" is used in ACMODE-ATM, only "CDP" is used in ACMODE-CDP, and only "SP" is used in ACMODE-SP. The experimental results of these three variants on CF6 and NCTP11 are summarized in Table 12. The best results are in bold.

Table 12 shows that ACMODE-SP obtains the best results on CF6. The results achieved by the other two variants have no significant difference. It can be observed from Figure 2a that "SP" is always in the dominant position during the entire evolutionary process, while other two CHTs play similar roles. This confirms that ACMODE allocates more computing resources to a better CHT. Table 12 reveals that ACMODE-CDP and ACMODE-ATM, respectively, get the best results in terms of *IGD* and *HV* on NCTP11. The performance of ACMODE-SP is the worst one among the three variants. Figure 2b indicates that "SP" is relatively stable during the whole evolutionary process. "ATM" plays an important role at the early and middle stages of evolution, while "CDP" is mainly used in ACMODE at the later stage of evolution. It can be seen from Figure 2 that the resources allocated to these three CHTs change in real time during evolution. It can be concluded that appropriate CHTs can be adaptively selected according to different types of CMOPs in different evolutionary periods.

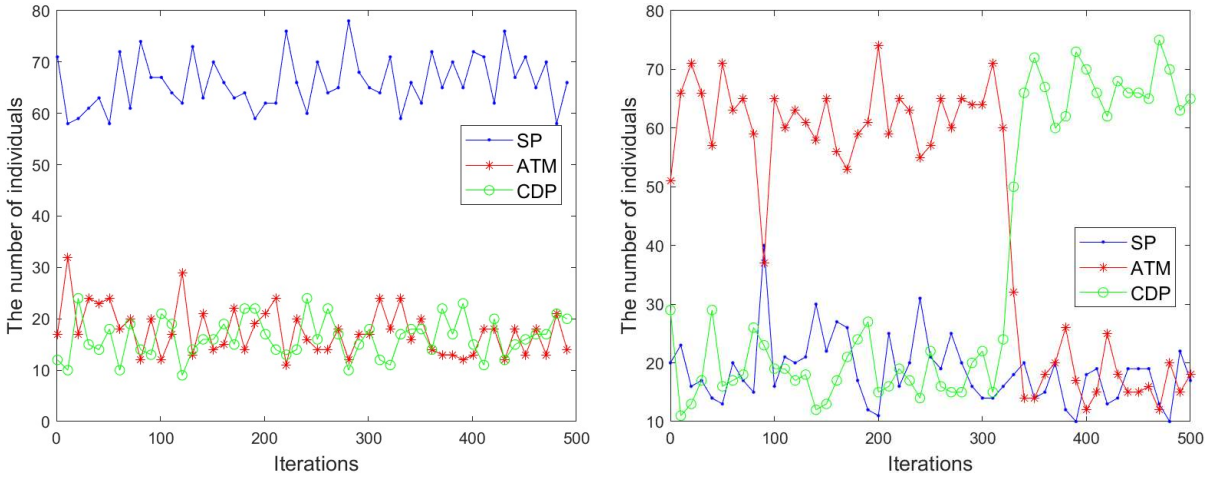


Figure 2. The number of individuals of CHTs changes during the whole evolutionary process. (**a**) CH6, (**b**) NCTP11.

Table 12. *IGD* and *HV* results of three variants on CF6 and NCTP11.

Function	Algorithm	*IGD*		*HV*	
		Mean	**Deviation**	**Mean**	**Deviation**
CF6	ACMODE-ATM	5.9609×10^{-2}	1.53×10^{-2}	6.1250×10^{-1}	1.16×10^{-2}
	ACMODE-CDP	5.7884×10^{-2}	1.77×10^{-2}	6.1339×10^{-1}	1.00×10^{-2}
	ACMODE-SP	$\mathbf{5.3083 \times 10^{-2}}$	$\mathbf{6.98 \times 10^{-3}}$	$\mathbf{6.2135 \times 10^{-1}}$	$\mathbf{6.30 \times 10^{-3}}$
NCTP11	ACMODE-ATM	5.2470×10^{-2}	1.11×10^{-2}	$\mathbf{5.9944 \times 10^{-1}}$	$\mathbf{3.58 \times 10^{-3}}$
	ACMODE-CDP	$\mathbf{4.9925 \times 10^{-2}}$	$\mathbf{1.00 \times 10^{-2}}$	5.9846×10^{-1}	4.36×10^{-3}
	ACMODE-SP	5.4795×10^{-2}	1.32×10^{-2}	5.9595×10^{-1}	3.50×10^{-3}

5.3.2. The Effectiveness of Adaptive Generation Strategy

To further validate the effectiveness of the proposed adaptive generation strategy, the evolution curves of generation strategies in two test functions (CF8 and LIR-CMOP9) are presented in this experiment.

Figure 3 shows that the number of the two-generation strategies changes during evolution. In this section, two variants are investigated by adopting different generation strategies. To be specific, only "DE/rand-to-best/1/bin" is used in ACMODE-best, and only "DE/current-to-rand/1" is used in ACMODE-current. The experimental results of these two variants on CF5 and LIR-CMOP9 are summarized in Table 13. The best results are in bold.

The results shown in Table 13 reveal that ACMODE-best performs better than ACMODE-current on CF5. As shown in Figure 3a, "DE/rand-to-best/1/bin" has good performance during the whole evolutionary process. ACMODE allocates more computing resources to "DE/rand-to-best/1/bin". This is consistent with the results shown in Table 13. Table 13 also demonstrates that ACMODE-best is slightly superior to ACMODE-current on LIR-CMOP9. It can be seen from Figure 3b that "DE/rand-to-best/1/bin" plays an important role at the very early stage, while "DE/current-to-rand/1" also displays good performance at the early stage. However, as evolution progresses, more computing resources are allocated to "DE/rand-to-best/1/bin" at later evolutionary stages. Figure 3 also indicates that the ACMODE can effectively allocate computing resources to generation strategies. It can be concluded that the ACMODE can adaptively select appropriate generation strategies at different evolutionary stages.

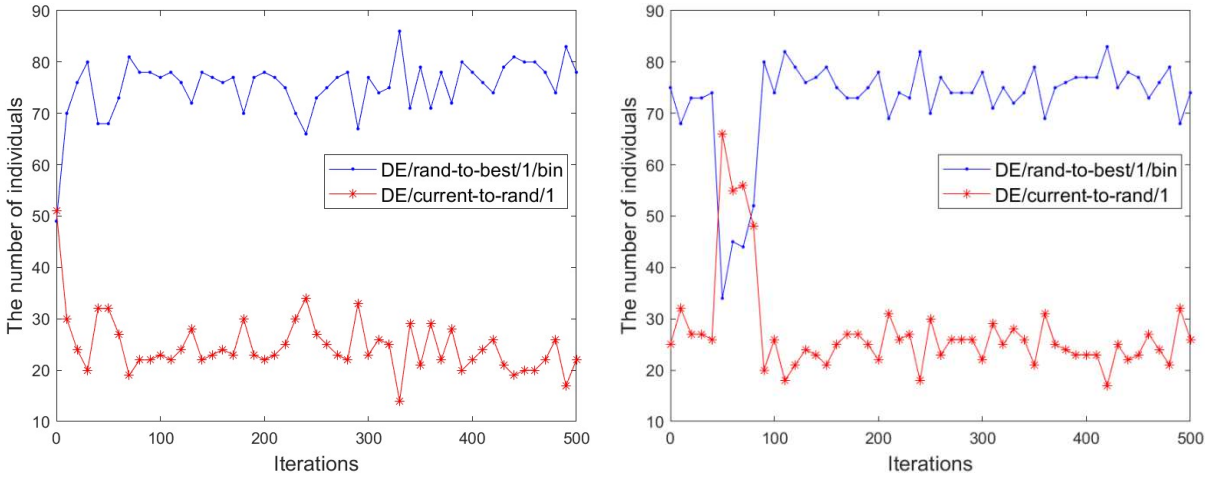


Figure 3. The number of individuals of two generation strategies changes during the whole evolutionary process. (**a**) CH5, (**b**) LIR-CMOP9.

Table 13. *IGD* and *HV* results of two variants on CF5 and LIR-CMOP9.

Function	Algorithm	*IGD*		*HV*	
		Mean	**Deviation**	**Mean**	**Deviation**
CF5	ACMODE-best	$\mathbf{2.5039 \times 10^{-1}}$	$\mathbf{1.34 \times 10^{-1}}$	$\mathbf{2.9561 \times 10^{-1}}$	$\mathbf{9.24 \times 10^{-2}}$
	ACMODE-current	2.9198×10^{-1}	1.37×10^{-1}	2.5686×10^{-1}	9.23×10^{-2}
LIR-CMOP9	ACMODE-best	$\mathbf{5.4438 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-2}}$	$\mathbf{3.1653 \times 10^{-1}}$	$\mathbf{2.02 \times 10^{-2}}$
	ACMODE-current	5.4840×10^{-1}	6.61×10^{-2}	3.1859×10^{-1}	3.98×10^{-2}

5.3.3. Visual Comparison on PF approximation

To testify the performance of the proposed algorithm in the objective space, three test functions (LIR-CMOP3, NCTP9 and MW8) are selected to provide the *PF* approximations obtained by ACMODE and its competitors. The results are given in Figure 4.

It can be observed from Figure 4a that the true *PF* of LIR-CMOP3 is discontinuous. LIR-CMOP3 exists infeasible barriers on the way to the true *PF*, which prevents algorithms from converging towards the true *PF*. Figure 4a also depicts that ACHT-CMODE, AGS-CMODE, MOEA/D-CDP and ANSGAIII only obtain part of the true *PF*. However, ACMODE is capable of passing through the infeasible region barriers to uniformly cover the true *PF*. Therefore, ACMODE performs better than the other four CMOEAs on LIR-CMOP3. Figure 4b shows that AGS-CMODE and ACHT-CMODE are slightly inferior to ACMODE on NCTP9. The final solutions obtained by MOEA/D-CDP are far away from the true *PF*. However, the solutions obtained by the proposed ACMODE are very close and evenly distributed to the whole true *PF*. It is worth noting that MW8 has three objectives. Figure 4c indicates that the final solutions obtained by ACHT-CMODE and AGS-CMODE are far away from the true *PF*. Moreover, MOEA/D-CDP and ANSGAIII are slightly inferior to ACMODE. The *PF* approximation obtained by ACMODE is evenly distributed and is very close to the true *PF*. therefore, it can be concluded that selecting a suitable CHT and generation strategies is effective and important in ACMODE.

5.3.4. Parameter Analysis

The probability ε is mainly used to balance the exploration and exploitation [50]. Therefore, the ε value plays a certain role in the ACMODE. The ε value in the specific algorithm is generally obtained by trial. In this experiment, all 65 test functions used in Section 5.2 are selected to investigate the sensitivity of ε. Moreover, ε is selected from the set {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.

Figure 5 provides the performance rankings of *HV* in terms of different ε values. A smaller average ranking value denotes better performance. It can be seen from Figure 5 that the overall performance of the algorithm is the best when ε = 0.5. Therefore, ε is set to 0.5 in the proposed algorithm.

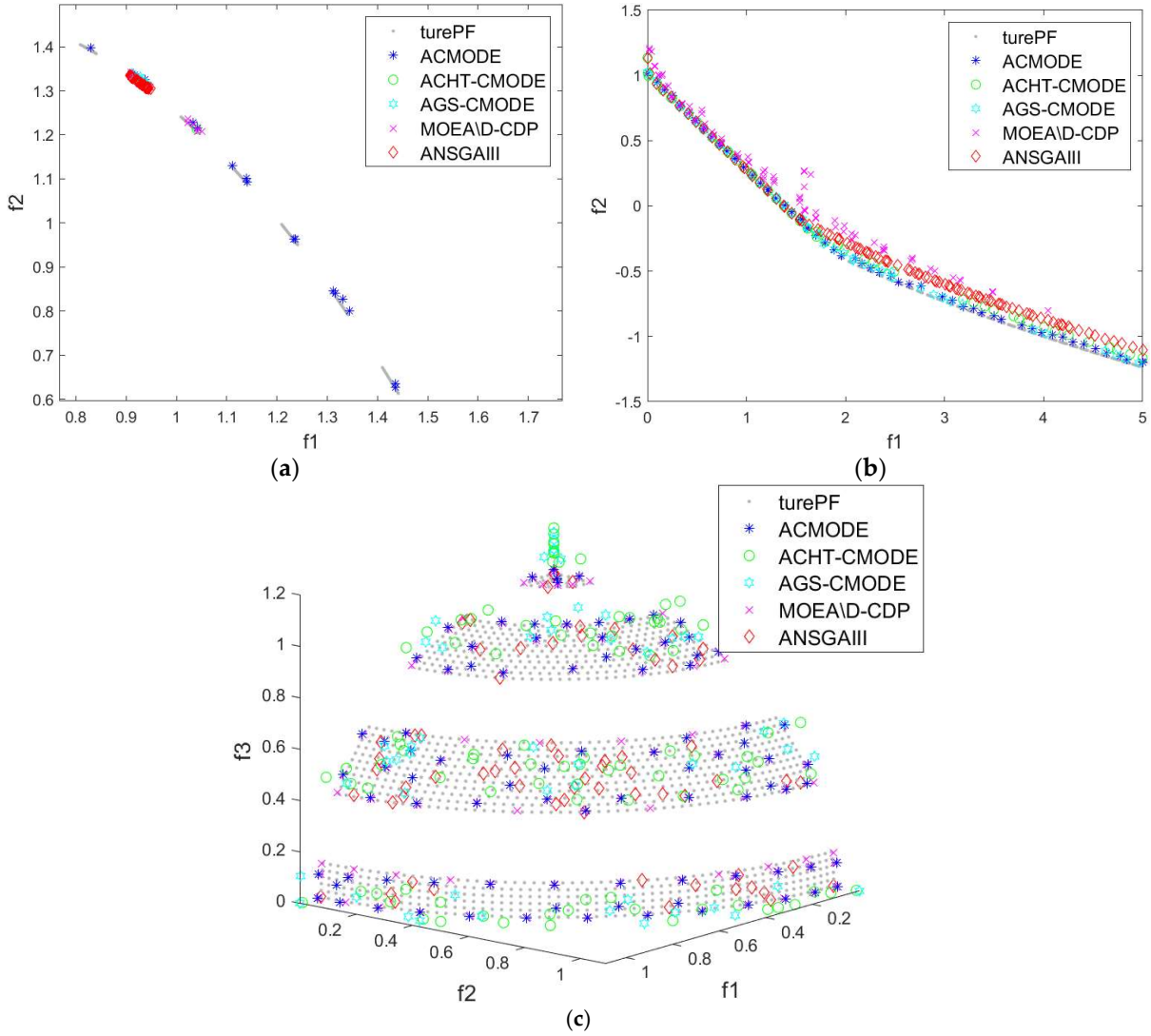


Figure 4. PF approximation of all compared algorithms. (**a**) LIR-CMOP3, (**b**) NCTP9, (**c**) MW8.

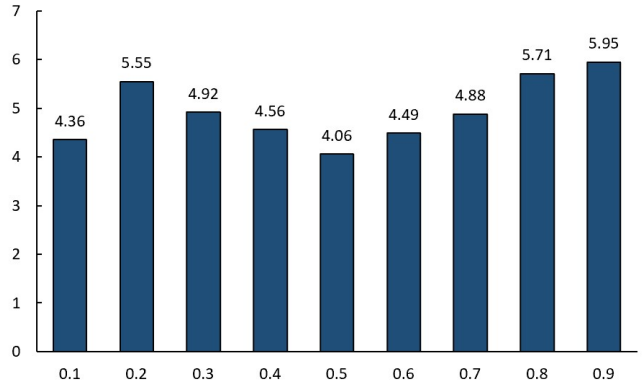


Figure 5. Performance ranking of *HV* with different ε values.

6. Discussion

From the above experimental comparisons and analyses, it can be seen that the overall performance of the proposed algorithm is better than that of four compared algorithms on five test suites. Specifically, the following results can be obtained.

- Compared with the AGS-CMODE, the experimental results show that using adaptive CHTs can assist the proposed algorithm in improving its performance on CMOPs. Moreover, using an adaptive generation strategy can help enhance the performance of the proposed algorithm when compared with the ACHT-CMODE. Therefore, it can be concluded that adaptive CHT and generation strategy is useful for ACOMDE to solve different types of CMOPs.
- MOEA/D-CDP works well on the CMOPs with a low feasibility ratio, and ANSGAIII is a self-adaptive evolutionary algorithm, in which reference points can adaptively update. Compared with these two algorithms, although they are effective on some specific CMOPs, the proposed algorithm outperforms them on most functions.
- The effectiveness of the proposed algorithm is also analyzed. The results demonstrate that the computational resources can be self-adaptively allocated to different CHTs and DE's generation strategies via the SARSA method during the entire evolutionary process.

However, ACMODE does not work very well when solving some DAS-CMOP series test problems with the difficulty of feasibility, convergence and diversity at the same time. The main reason may be that the existing CHTs and generation strategies may not enable the proposed algorithm to find feasible solutions. Moreover, the adaptive process may waste some computational resources, thus the efficiency of learning method is important.

7. Conclusions

CHTs and generation strategies significantly affect the performance of CMOEAs. In the present work, an adaptive constrained multi-objective differential evolution algorithm based on state–action–reward–state–action approach (ACMODE) is introduced to implement adaptation of CHTs and generation strategies, which can be automatically selected via a SARSA method. The performance of the ACMODE is compared with four other CMOEAs on five test suites, and the experimental results demonstrate that ACMODE is competitive in handling CMOPs. The main reason is that the ACMODE can adaptively select the appropriate CHT and generation strategy at different evolutionary stages when solving different types of CMOPs. Finally, the effectiveness of the introduced components of ACMODE is also demonstrated. Although adaptive adjustment of CHTs and generation strategies have been carried out in a current study, the selected CHT and generation strategy also have a great influence on the performance of the proposed algorithm. In future work, we will explore more novel CHTs and generation strategies with good performance, and incorporated them into ACMODE to solve CMOPs. Moreover, we will apply ACMODE to real-world problems to further confirm its effectiveness.

Author Contributions: Methodology: C.C., Q.L., Q.F.; writing—original draft: C.C., Q.L., Q.F.; writing—review and editing: Q.L., Q.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the National Natural Science Foundation of China (No. 61603244, 71904116), the National Social Science Fund of China (No. 18BGL103), and Shanghai Science and Technology Commission (No. 19DZ1209600, No. 18DZ1201500).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The original version of this paper was presented at the 16th International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA 2021), December 2021. This paper was recommended for publication in revised form by the BIC-TA 2021 conference committees.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Maminov, A.; Posypkin, M. Constrained Multi-objective Robot's Design Optimization. In Proceedings of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg, Russia, 27–30 January 2020; pp. 1992–1995.
2. Liu, J.; Yang, Y.; Tan, S.; Wang, H. Application of Constrained Multi-objective Evolutionary Algorithm in a Compressed-air Station Scheduling Problem. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 2023–2028.
3. Li, B.; Wang, J.; Xia, N. Dynamic Optimal Scheduling of Microgrid Based on ε constraint multi-objective Biogeography-based Optimization Algorithm. In Proceedings of the 2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE), Dalian, China, 19–20 September 2020; pp. 389–393.
4. Wang, J.; Li, Y.; Zhang, Q.; Zhang, Z.; Gao, S. Cooperative Multiobjective Evolutionary Algorithm With Propulsive Population for Constrained Multiobjective Optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, 1–16. [CrossRef]
5. Datta, R.; Deb, K.; Segev, A. A bi-objective hybrid constrained optimization (HyCon) method using a multi-objective and penalty function approach. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia-San Sebastián, Spain, 5–8 June 2017; pp. 317–324.
6. Yuan, J.; Liu, H.L.; Ong, Y.S.; He, Z. Indicator-based Evolutionary Algorithm for Solving Constrained Multi-objective Optimization Problems. *IEEE Trans. Evol. Comput.* **2021**, 1. [CrossRef]
7. Cui, C.X.; Fan, Q.Q. Constrained Multi-objective Differential Evolutionary Algorithm with Adaptive Constraint Handling Technique. *World Sci. Res. J.* **2021**, *7*, 322–339. [CrossRef]
8. Richard, S.S.; Andrew, G.B. Temporal-Difference Learning. In *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998; pp. 133–160.
9. Yu, X.B.; Lu, Y.Q. A corner point-based algorithm to solve constrained multi-objective optimization problems. *Appl. Intell.* **2018**, *48*, 3019–3037. [CrossRef]
10. Xiang, Y.; Yang, X.W.; Huang, H.; Wang, J.H. Balancing Constraints and Objectives by Considering Problem Types in Constrained Multiobjective Optimization. *IEEE Trans. Cybern.* **2021**, 1–14. [CrossRef]
11. Fan, Z.; Li, W.J.; Cai, X.Y.; Li, H.; Wei, C.M.; Zhang, Q.F.; Deb, K.; Goodman, E. Push and pull search for solving constrained multi-objective optimization problems. *Swarm Evol. Comput.* **2019**, *44*, 665–679. [CrossRef]
12. Uribe, L.; Lara, A.; Deb, K.; Schutze, O. A new gradient free local search mechanism for constrained multi-objective optimization problems. *Swarm Evol. Comput.* **2021**, *67*, 100938. [CrossRef]
13. Liu, Z.Z.; Wang, Y.; Wang, B.C. Indicator-Based Constrained Multiobjective Evolutionary Algorithms. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 5414–5426. [CrossRef]
14. Tian, Y.; Zhang, T.; Xiao, J.; Zhang, X.; Jin, Y. A Coevolutionary Framework for Constrained Multiobjective Optimization Problems. *IEEE Trans. Evol. Comput.* **2021**, *25*, 102–116. [CrossRef]
15. Liu, Z.Z.; Wang, Y. Handling Constrained Multiobjective Optimization Problems With Constraints in Both the Decision and Objective Spaces. *IEEE Trans. Evol. Comput.* **2019**, *23*, 870–884. [CrossRef]
16. Ming, M.; Wang, R.; Ishibuchi, H.; Zhang, T. A Novel Dual-Stage Dual-Population Evolutionary Algorithm for Constrained Multi-Objective Optimization. *IEEE Trans. Evol. Comput.* **2021**, 1. [CrossRef]
17. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
18. Xu, B.; Duan, W.; Zhang, H.F.; Li, Z.Q. Differential evolution with infeasible-guiding mutation operators for constrained multi-objective optimization. *Appl. Intell.* **2020**, *50*, 4459–4481. [CrossRef]
19. Yu, K.; Liang, J.; Qu, B.; Luo, Y.; Yue, C. Dynamic Selection Preference-Assisted Constrained Multiobjective Differential Evolution. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, 1–12. [CrossRef]
20. Yang, Y.K.; Liu, J.C.; Tan, S.B. A partition-based constrained multi-objective evolutionary algorithm. *Swarm Evol. Comput.* **2021**, *66*, 100940. [CrossRef]
21. Lin, Y.; Du, W.; Du, W. Multi-objective differential evolution with dynamic hybrid constraint handling mechanism. *Soft Comput.* **2019**, *23*, 4341–4355. [CrossRef]
22. Yu, X.B.; Yu, X.R.; Lu, Y.Q.; Yen, G.G.; Cai, M. Differential evolution mutation operators for constrained multi-objective optimization. *Appl. Soft Comput.* **2018**, *67*, 452–466. [CrossRef]
23. Wang, J.; Liang, G.; Zhang, J. Cooperative Differential Evolution Framework for Constrained Multiobjective Optimization. *IEEE Trans. Cybern.* **2019**, *49*, 2060–2072. [CrossRef]
24. Moniz, N.; Monteiro, H. No Free Lunch in imbalanced learning. *Knowl.-Based Syst.* **2021**, *227*, 107222. [CrossRef]

25. Yang, Y.K.; Liu, J.C.; Tan, S.B.; Wang, H.H. A multi-objective differential evolutionary algorithm for constrained multi-objective optimization problems with low feasible ratio. *Appl. Soft Comput.* **2019**, *80*, 42–56. [CrossRef]
26. Yang, N.; Liu, H.L. Adaptively Allocating Constraint-Handling Techniques for Constrained Multi-objective Optimization Problems. *Int. J. Pattern Recognit. Artif. Intell.* **2021**, *35*, 2159032. [CrossRef]
27. Liu, B.J.; Bi, X.J. Adaptive ε-Constraint Multi-Objective Evolutionary Algorithm Based on Decomposition and Differential Evolution. *IEEE Access* **2021**, *9*, 17596–17609. [CrossRef]
28. Mashwani, W.K.; Salhi, A.; Yeniay, O.; Jan, M.A.; Khanum, R.A. Hybrid adaptive evolutionary algorithm based on decomposition. *Appl. Soft Comput.* **2017**, *57*, 363–378. [CrossRef]
29. Zhang, L.; Bi, X.J.; Wang, Y.J. Adaptive Truncation technique for Constrained Multi-Objective Optimization. *Ksii Trans. Internet Inf. Syst.* **2019**, *13*, 5489–5511. [CrossRef]
30. Samanipour, F.; Jelovica, J. Adaptive repair method for constraint handling in multi-objective genetic algorithm based on relationship between constraints and variables. *Appl. Soft Comput.* **2020**, *90*, 106143. [CrossRef]
31. Fan, Q.; Zhang, Y.; Li, N. An Autoselection Strategy of Multiobjective Evolutionary Algorithms Based on Performance Indicator and Its Application. *IEEE Trans. Autom. Sci. Eng.* **2021**, 1–15. [CrossRef]
32. Woldesenbet, Y.G.; Yen, G.G.; Tessema, B.G. Constraint Handling in Multiobjective Evolutionary Optimization. *IEEE Trans. Evol. Comput.* **2009**, *13*, 514–525. [CrossRef]
33. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
34. Wang, Y.; Cai, Z.; Zhou, Y.; Zeng, W. An Adaptive Tradeoff Model for Constrained Evolutionary Optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 80–92. [CrossRef]
35. Bosman, P.A.N.; Thierens, D. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Trans. Evol. Comput.* **2003**, *7*, 174–188. [CrossRef]
36. Yuan, J.; Liu, H.-L.; He, Z. A constrained multi-objective evolutionary algorithm using valuable infeasible solutions. *Swarm Evol. Comput.* **2022**, *68*, 101020. [CrossRef]
37. Zitzler, E.; Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* **1999**, *3*, 257–271. [CrossRef]
38. Fan, Q.; Yan, X.; Zhang, Y.; Zhu, C. A Variable Search Space Strategy Based on Sequential Trust Region Determination Technique. *IEEE Trans. Cybern.* **2021**, *51*, 2712–2724. [CrossRef] [PubMed]
39. Fan, Q.Q.; Wang, W.L.; Yan, X.F. Multi-objective differential evolution with performance-metric-based self-adaptive mutation operator for chemical and qbiochemical dynamic optimization problems. *Appl. Soft Comput.* **2017**, *59*, 33–44. [CrossRef]
40. Shahrabi, J.; Adibi, M.A.; Mahootchi, M. A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Comput. Ind. Eng.* **2017**, *110*, 75–82. [CrossRef]
41. Jan, M.A.; Khanum, R.A. A study of two penalty-parameterless constraint handling techniques in the framework of MOEA/D. *Appl. Soft Comput.* **2013**, *13*, 128–148. [CrossRef]
42. Jain, H.; Deb, K. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Trans. Evol. Comput.* **2014**, *18*, 602–622. [CrossRef]
43. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80–83. [CrossRef]
44. Friedman, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *J. Am. Stat. Assoc.* **1937**, *32*, 675–701. [CrossRef]
45. Zhang, Q.; Zhou, A.; Zhao, S.; Suganthan, P.N.; Tiwari, S. Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition. *Mech. Eng.* **2008**, *264*, 1–30.
46. Fan, Z.; Fang, Y.; Li, W.; Cai, X.; Wei, C.; Goodman, E. MOEA/D with angle-based constrained dominance principle for constrained multi-objective optimization problems. *Appl. Soft Comput. J.* **2018**, *74*, 621–633. [CrossRef]
47. Li, J.P.; Wang, Y.; Yang, S.; Cai, Z. A comparative study of constraint-handling techniques in evolutionary constrained multiobjective optimization. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4175–4182.
48. Ma, Z.; Wang, Y. Evolutionary Constrained Multiobjective Optimization: Test Suite Construction and Performance Comparisons. *IEEE Trans. Evol. Comput.* **2019**, *23*, 972–986. [CrossRef]
49. Fan, Z.; Li, W.J.; Cai, X.Y.; Li, H.; Wei, C.M.; Zhang, Q.F.; Deb, K.; Goodman, E. Difficulty Adjustable and Scalable Constrained Multiobjective Test Problem Toolkit. *Evol. Comput.* **2020**, *28*, 339–378. [CrossRef] [PubMed]
50. Liu, Y.; Cao, B.; Li, H. Improving ant colony optimization algorithm with epsilon greedy and Levy flight. *Complex Intell. Syst.* **2021**, *7*, 1711–1722. [CrossRef]