mathematics

MDPI

Article

Dynamic Constrained Boundary Method for Constrained Multi-Objective Optimization

Qiuzhen Wang [1,2], Zhibing Liang [1,2,*], Juan Zou [1,2], Xiangdong Yin [3], Yuan Liu [1,2], Yaru Hu [1,2] and Yizhang Xia [1,2]

[1] Key Laboratory of Intelligent Computing and Information Processing, Ministry of Education, School of Computer Science and School of Cyberspace Science of Xiangtan University, Xiangtan 411100, China
[2] Faculty of School of Computer Science and School of Cyberspace Science of Xiangtan University, Xiangtan 411105, China
[3] Faculty of Informational Engineering, Hunan University of Science and Engineering, Yongzhou 411201, China
* Correspondence: 202021002433@smail.xtu.edu.cn; Tel.: +86-15197866538

Abstract: When solving complex constrained problems, how to efficiently utilize promising infeasible solutions is an essential issue because these promising infeasible solutions can significantly improve the diversity of algorithms. However, most existing constrained multi-objective evolutionary algorithms (CMOEAs) do not fully exploit these promising infeasible solutions. In order to solve this problem, a constrained multi-objective optimization evolutionary algorithm based on the dynamic constraint boundary method is proposed (CDCBM). The proposed algorithm continuously searches for promising infeasible solutions between UPF (the unconstrained Pareto front) and CPF (the constrained Pareto front) during the evolution process by the dynamically changing auxiliary population of the constraint boundary, which continuously provides supplementary evolutionary directions to the main population and improves the convergence and diversity of the main population. Extensive experiments on three well-known test suites and three real-world constrained multi-objective optimization problems demonstrate that CDCBM is more competitive than seven state-of-the-art CMOEAs.

Keywords: constrained multi-objective optimization problems (CMOPs); dynamic constrained boundary

MSC: 68W50; 68T05

Citation: Wang, Q.; Liang, Z.; Zou, J.; Yin, X.; Liu, Y.; Hu, Y.; Xia, Y. Dynamic Constrained Boundary Method for Constrained Multi-Objective Optimization. *Mathematics* **2022**, *10*, 4459. https://doi.org/10.3390/math10234459

Academic Editor: Xavier Blasco

Received: 30 September 2022
Accepted: 20 November 2022
Published: 26 November 2022

1. Introduction

Practical applications often involve constrained multi-objective optimization problems, which usually contain multiple conflicting objectives and constraints. Without loss of generality, minimizing constrained multi-objective optimization problems (CMOPs) can be formulated as follows [1]:

$$\begin{cases} minimize\ F(x) = (f_1(x), f_2(x), f_3(x), \ldots, f_m(x)), x \in \Omega \\ g_j(x) \le 0, j = 1, \ldots, p \\ h_j(x) = 0, j = 1, \ldots, q \end{cases} \quad (1)$$

where $F(x)$ is an objective function that contains m objectives and $x = (x_1, x_2, \ldots, x_d)$ is a d-dimension solution in the decision space Ω. $g_j(x)$ and $h_j(x)$ are inequalities and equality constraints whose quantities are p and q, respectively. Usually, the constraint violation of x for the jth constraint is calculated as follows [2]:

$$CV_j(x) = \begin{cases} max\{0, g_j(x)\}, & j = 1, \ldots, p \\ \max\{0, |h_j(x)| - \delta\}, & j = p+1, \ldots, q \end{cases} \quad (2)$$

where δ is a positive constraint boundary relaxation factor (usually taken as 1e-6). The overall constraint violation value of x (CV) can be summarized as:

$$CV(x) = \sum_{j=1}^{p+q} CV_j(x) \tag{3}$$

The feasibility of a solution can be determined based on the constraint total violation CV. If the overall constraint violation (CV(x)) of one solution is zero, then it is called a feasible solution; otherwise, it is an infeasible solution. Given two feasible solutions x_1 and $x_2 \in \Omega$, x_1 is said to dominate x_2 if $f_i(x_1) \leq f_i(x_2)$ for every $i \in \{1, \ldots, m\}$ and $f_j(x_1) \prec f_j(x_2)$ for at least one $j \in \{1, \ldots, m\}$, denoted as $x_1 \prec x_2$. When a solution is not dominated by any other feasible solutions, the solution is called a feasible non-dominated solution or Pareto optimal solution. In the objective space, all Pareto optimal solutions are called the Pareto optimal solution set (PS), and the mapping of the Pareto optimal solution set in the objective space is called the Constrained Pareto Front (PF). The key to solving CMOPs is finding a set of feasible solutions with well-convergence (find Pareto optimal fronts solutions as many as possible) and well-distribution (found the solution covers all PF surfaces as many as possible) to approximate the PF [3]. For CMOPs, the objective space contains two Pareto optimal fronts, depending on whether the constraints are considered or not. In order to distinguish between them, the constrained Pareto front (CPF) and the unconstrained Pareto front (UPF) are usually used to represent the PF of the constrained multi-objective optimization problem and the unconstrained multi-objective optimization problem, respectively. Earlier researchers classified CMOPs into four categories based on the relationship between UPF and CPF [4]. These categories are (1) type-I: CPF is the same as UPF, (2) type-II: CPF is part of UPF, (3) type-III: CPF and UPF partially overlap, and (4) type-IV: CPF is wholly separated from UPF.

The challenge of constrained multi-objective evolutionary algorithms (CMOEAs) in solving CMOPs is to maintain a better balance between objective optimization and constraint satisfaction to achieve better convergence [5]. For conflicting objectives and constraints, if constraints are prioritized more than objectives, the population may quickly fall into the local feasible regions, unable to approach the true PF (especially when large infeasible regions block feasible areas). On the other hand, if objectives are prioritized more than constraints, the exploration ability of the solution is improved to a certain extent, but the convergence quality of the population may be reduced. Consequently, balancing the conflict between objectives and constraints in the evolutionary process is crucial.

Among the algorithms for solving CMOPs, the most widely used and the most typical category is the Constrained Dominance Principle (CDP) algorithm [6]. The CDP algorithm pays more attention to the feasibility of the solution (constraints have higher priority) and prefers to choose feasible solutions over infeasible solutions. Therefore, when a large infeasible area blocks the feasible area, the convergence performance of this kind of algorithm decreases. Some existing dual-population optimization algorithms cannot effectively solve CMOPs [7,8] whose optimal region is far from UPF because the auxiliary population cannot continuously maintain and utilize promising infeasible solutions during the evolution process. Furthermore, complex constraints lead to large infeasible regions between feasible regions, and small and discontinuous feasible regions make it difficult to search the population. Therefore, it is necessary to use promising infeasible solutions ($CV < \varepsilon$) reasonably and continuously in the evolution process to help the main population pass through the infeasible region and search for a narrow potential feasible region.

Following this idea, this paper proposes dual-population based evolutionary algorithm CDCBM:

1. An auxiliary population state detection strategy is proposed, which can detect the relationship between the auxiliary population and the boundary of the feasible region. According to the detected results, the auxiliary population is effectively selected in

the evolution process, and more useful objective information is provided for the main population.

2. For the auxiliary population, a dynamic constraint boundary method is proposed, which uses various promising infeasible solutions ($CV < \varepsilon$) to break through the infeasible obstacles, helps the population to overcome the obstacles in the infeasible area, and makes full use of the infeasible areas approaching the feasible area. The favorable information about the solution can be used to search for potential feasible regions.

2. Materials and Methods

2.1. Related Work

This section introduces existing CMOEAs with constraint handling techniques (CHT) [9].

Early MOEAs solved CMOPs based on feasibility. For example, NSGA-II [6] is one of the most representative CDP-embedded algorithms. In the CDP method, when individuals A and B are both feasible solutions, the one with a smaller objective value enters the next generation. For a feasible solution A and an infeasible solution B, the feasible solution A is better than the infeasible solution B. For the two infeasible solutions A and B, the individual with a smaller CV value is selected. In other words, the method prefers feasible solutions to infeasible solutions. When CMOPs have discrete feasible regions or infeasible obstacles, NSGA-II [6] quickly converges to local feasible regions. To address this problem, ToR [10] uses a dual-rank fitness function that combines the CDP and Pareto strengths to evaluate the fitness of each solution by the weighted sum of the two ranks. However, the weights it constrains are always higher than the objective, which may lead to insufficient diversity. In [11], the decomposition-based MOEA (MOEA/D) and CDP were combined to form the MOEA/D-CDP method. Fan et al. [12] designed a novel angle-based CDP (ACDP) for CMOPs, which guides environment selection by measuring the angle between the parent and the corresponding child, further exploiting the information of the infeasible solutions. However, it is difficult for this method to achieve a good balance between objectives and constraints, which inevitably leads to premature convergence.

Several researchers have developed multistage CMOEAs to balance objective and constraint satisfaction. The Push-Pull Strategy (PPS) [13] is representative of a two-stage algorithm, where the goal of the push stage is to reach the UPF across infeasible regions. In the pull phase, the population is updated considering all constraints so that the population converges to the CPF. Similarly, Liu et al. [14] proposed a two-stage framework (ToP), where the first stage mainly looks for promising feasible regions, and the second stage looks for the CPF by strengthening convergence. However, there are certain difficulties in converting the first stage to the second stage. The MSCMO [15] algorithm gradually adds constraints during evolution, maintaining the diversity of the population by using the solutions found in the previous stage. Zhu et al. [16] developed an algorithm based on a detection-escape strategy, which guides the population search and escapes the stagnation state by adjusting the weight of CVs when detecting that the population search stagnates and falls into a local optimum. Tian et al. [5] adjusted the search behavior of the population according to the proportion of feasible solutions. Objective and constraint satisfaction are considered equally important in evolution when the proportion of feasible solutions for the population is less than λ. When the feasible solution is larger than λ, the constraint is given higher priority.

Some multiple population CMOEAs also involve resolving the conflict between goals and constraint satisfaction. For example, Li et al. [8] designed a dual-archive evolutionary algorithm (C-TAEA), which includes a convergence-oriented archive (CA) and a diversity-oriented archive (DA). The CA considers both objectives and constraints and mainly tends to approximate the CPF. The DA does not consider any constraints and aims to explore areas underutilized by the CA. Most of the offspring generated by the population is in the region between the UPF and CPF, which has poor convergence and diversity. Tian et al. [7] proposed a co-evolutionary framework for constrained multi-objective optimization (CCMO),

where the first population is used to search for the CPF, and the second population ignores constraints to search for the UPF. The algorithm achieves better results on CMOPs with a high correlation between the CPF and UPF but poor performance on CMOPs when the UPF and CPF are far from each other. Ming et al. [17] proposed a DD-CMOEA algorithm with dual phases (i.e., exploration and exploitation) and dual population characteristics. The main and auxiliary populations are responsible for exploring feasible and infeasible solutions. Although the promising infeasible solutions obtained by the auxiliary population can help the main population to converge better, there is a lack of search for some parts of the PF, resulting in not-so-good performance. c-DPEA [18] is a co-evolutionary algorithm in which two populations use different methods to deal with infeasible solutions. Population 1 and Population 2 adopt a new adaptive penalty function and feasibility-oriented method to deal with infeasible solutions, respectively. However, the latter stages of Population 2 do not provide sufficiently diverse solutions for Population 1. Zou et al. [19] proposed a dual-population algorithm based on alternative evolution and degeneration for solving CMOPs. It can make good use of its secondary population and alternative between evolution and degeneration according to the state of the secondary population to provide good information about the secondary population to the main population. In EMCMO [20], the optimization of traditional CMOPs is transformed into two related tasks: one task targets the original CPF, and the other task ignores all constraints and only considers the objective. The responsibility of the second task is mainly to continuously provide helpful knowledge of the target to the first task, thereby facilitating the resolution of CMOPs.

Although these multiple population CMOEAs utilize infeasible solutions to maintain the diversity of the main population while exhibiting good performance on some CMOPs, there are still certain difficulties in solving complex constrained problems. The main reason is that the auxiliary population cannot continuously provide effective information during the evolution process. Aiming at this problem, we propose a method of dynamically constrained boundaries, which can continuously provide promising infeasible solutions for the main population and improve its convergence and diversity.

2.2. Methods

As shown in Table 1, all the variables used by the algorithm are presented.

Table 1. Variables used in the algorithm.

Variable	Description
P1	main population
P2	auxiliary populations
ε_0	ε initial value
O1	offspring of *P1*
O2	offspring of *P2*
obj_k	sum of overall objective values of k generation in *P2*
σ	$\lvert obj_k - obj_{k-1} \rvert$
ε_k	ε value before updated
Nc	proportion of nondominated solutions in *P2*
f	the ratio of solutions (CV<=0) in *P2*
a	constraint tolerance
TrP1	temporary population of *P1*
TrP2	temporary population of *P2*

Algorithm 1 gives the pseudo-code of CDCBM (a constrained multi-objective optimization evolutionary algorithm based on the dynamic constraint boundary method). First,

two initial populations, *P1* and *P2*, are randomly generated in the search space, and each population has N individuals. Then, in lines 5–6 of the algorithm, the binary competition selection method is used to select from select mating parent sets in *P1* and *P2*. Then new solutions are generated by simulating binary crossover [21] and polynomial mutation [22], denoted as *O1* and *O2*, respectively. We can see no interaction between the two populations during the initialization phase to generate offspring. Then, we need to detect whether *P2* has reached UPF. Since *P2* does not consider any constraint, when the sum of the current objective values of the population differs little from the previous generation, *P2* enters the steady state, which can also be said to reach UPF. Therefore, we introduced a minimal parameter σ (1e-3 in this paper). And σ is the difference between the sum of all individual objective values of the current generation and that of the previous generation. If the σ is less than the set threshold, there is no dominant solution in *P2*. At this time, the ε value is infinite, so it can be considered that *P2* reaches the UPF. When *P2* reaches the UPF, the algorithm goes to lines 9–10 and sets the ε value to be the average of the constraint violation values in *P2*. Then, calculate the ratio f (the ratio of solutions (CV<=0) in *P2*). When f = 1, all the individuals in *P2* are feasible solutions. It means that all the UPFs in *P2* are at the boundary of the feasible region. Thus, we judge the relationship between the auxiliary population and the boundary of the feasible region by the results of comparing f with λ (1 in this case).

Algorithm 1: Procedure of CDCBM

Input: N (population size), obj_k (sum of overall objective values of k generation in P2), Nc (proportion of nondominated solutions in P2);
Output: P1(final population);

1. P1← Randomly generate N individuals;
2. P2← Randomly generate N individuals;
3. ε_0 = inf;
4. while Termination conditions are not satisfied do
5. Select mating parents from P1 and generate N offspring denoted as O1;
6. Select mating parents from P2 and generate N offspring denoted as O2;
7. $\sigma = |obj_k - obj_{k-1}|$;
8. if σ <∇ and Nc == 1 then
9. ε_k = Mean CV value in P2;
10. f: the ratio of solutions (CV<=0) in P2;
11. end
12. if $\varepsilon_k > \varepsilon_{threshold}$ then
13. $\varepsilon_k = (1 - \tau) * \varepsilon_k$;
14. else
15. $\varepsilon_k = a$;
16. end
17. if f is smaller than λ
18. P1 ← P1 ∪ O1 ∪ O2;
19. P2 ← P2 ∪ O2 ∪ O1;
20. P1 ← Use the CDP method to select N individuals;
21. P2 ← Use the ε method to select N individuals;
22. end
23. else
24. Algorithm 2;
25. end
26. end

Algorithm 1 gives the pseudo-code of CDCBM (a constrained multi-objective optimization evolutionary algorithm based on the dynamic constraint boundary method). First, two initial populations, *P1* and *P2*, are randomly generated in the search space, and each population has N individuals. Then, in lines 5–6 of the algorithm, the binary competition selection method is used to select from select mating parent sets in *P1* and *P2*. Then new solutions are generated by simulating binary crossover [21] and polynomial mutation [22],

denoted as *O1* and *O2*, respectively. We can see no interaction between the two populations during the initialization phase to generate offspring. Then, we need to detect whether *P2* has reached UPF. Since *P2* does not consider any constraint, when the sum of the current objective values of the population differs little from the previous generation, *P2* enters the steady state, which can also be said to reach UPF. Therefore, we introduced a minimal parameter σ (1e-3 in this paper); σ is the difference between the sum of all individual objective values of the current generation and that of the previous generation. If the σ is less than the set threshold, there is no dominant solution in *P2*. At this time, the ε value is infinite, so it can be considered that *P2* reaches the UPF. When *P2* reaches the UPF, the algorithm goes to lines 9–10 and sets the ε value to be the average of the constraint violation values in *P2*. Then, calculate the ratio f (the ratio of solutions (CV<=0) in *P2*). When f = 1, all the individuals in *P2* are feasible solutions. It means that all the UPFs in *P2* are at the boundary of the feasible region. Thus, we judge the relationship between the auxiliary population and the boundary of the feasible region by the results of comparing f with λ (1 in this case).

In lines 12–16 of the algorithm, after *P2* converges to UPF, the ε value is gradually reduced by the descending speed control parameter τ [13], τ is taken as 0.05 in this paper. ε updates are critical to the environmental selection of *P2*. When ε decreases to a minimal threshold (1e-4 in this paper), we use a modified sigmoid function to update ε. a is also declining with evolutionary generations, and $a \in [0, 1]$. In this step, we update the value of ε back and forth between 0 and a to maintain the promising infeasible solutions in *P2* and search for potentially feasible regions. The calculation of the constraint tolerance a is as follows:

$$a = 2 - \frac{2}{1 + e^{-10T/T_{max}}} \tag{4}$$

where T is the current generation and T_{max} (the setting in this paper is 300,000) is the maximal generation. When the value of ε is less than its minimum threshold (1e-4 in this paper), it means that the CV of the *P2* is close to 0. At this time, by gradually reducing the constraint tolerance with the evolutionary generation, P2 retains some infeasible solutions with good objective values. With this operation, *P2* searches back and forth between UPF and CPF to advance the population to the feasible region.

Inspired by [20], in the evolutionary process, selecting appropriate parental individuals and offspring individuals to enter the population for the next update can effectively save the number of evaluations. Therefore, we improve on [20] by designing an auxiliary population state detection strategy. In [20] the evolution process is simply divided into two stages, the first stage accounts for one-fifth of the entire evolution process. On the contrary, we approximate the relationship between UPF and CPF by calculating the ratio f of individuals after the auxiliary population converges to UPF. If f is equals to λ (1 in this case), the algorithm executes lines 18–21; otherwise, it enters Algorithm 2. When f is equals to λ, it indicating that the auxiliary population has entered a stable state. At this time, all solutions in the auxiliary population are feasible solutions. It is approximately determined that the UPF and the CPF coincide. After estimating the state of the auxiliary populations, the offspring *O1* and *O2* generated by the population are merged into the two parent generations for environmental selection. *P1* uses the CDP method for environment selection, and *P2* uses the ε method to select a new P2. In fact, at this point, $\varepsilon = 0$, which has the same effect as the CDP method. If f is not equal to λ, enter Algorithm 2. At this time, since the solutions on the CPF are feasible solutions, while most of the solutions of the auxiliary population are infeasible solutions. It is approximately judged that the UPF and the CPF are wholly or partially separated.

In lines 1–4 of Algorithm 2, *P1* and O1, and *P2* and *O2* are merged, respectively, to obtain a new population with 2N individuals $P1'$ and $P2'$. Next, $P1'$ uses the ε method to select the best N individuals, and $P2'$ uses the CDP method to select. In this step, we want to select the more suitable parent or offspring from $P1'$ to participate in the interaction with *P2*, so ε method is chosen; we want to select the more suitable parent or offspring from

$P2'$ to participate in the interaction with $P1$, Thus the CDP method is chosen. Then, the success rate of the parent and offspring populations entering the next iteration is calculated according to Equations (5) and (6). The success rate of parent P_j and child O_j is calculated as follows:

$$\alpha P_j = \frac{n\_P_j}{N}, j = 1, 2 \tag{5}$$

$$\alpha O_j = \frac{n\_O_j}{N}, j = 1, 2 \tag{6}$$

Algorithm 2: Procedure of CDCBM

Input: N (population size), P1, P2;
Output: P1;

1. $P1' \leftarrow$ P1 $\cup$ O1;
2. $P2' \leftarrow$ P2 $\cup$ O2;
3. Evaluate $P1'$ on the ε method;
4. Evaluate $P2'$ on o the CDP method;
5. TrP1 $\leftarrow$ Based on Equations (5) and (6);
6. TrP2 $\leftarrow$ Based on Equations (5) and (6);
7. P1 $\leftarrow$ P1 $\cup$ O1 $\cup$ TrP2;
8. P2 $\leftarrow$ P2 $\cup$ O2 $\cup$ TrP1;
9. P1 $\leftarrow$ Select N individuals from P1 based on the CDP method;
10. P2 $\leftarrow$ Select N individuals from P2 based on the ε method;
11. end

Among them, αP_j and αO_j represent the success rates of P_j and O_j, respectively; n_P_j and n_O_j are the numbers of parents and offspring in the best N individuals. When j = 1, if $\alpha P_1 < \alpha O_1$, it means that the offspring is more suitable than the parent to be selected for the next evolutionary update. In this case, $O1$ is kept in the temporary population $TrP2$. Otherwise, $P1$ is the better choice. The operation of j = 2 is the same as the operation of j = 1. Finally, the CDP method and the ε method are used to select the environment for $P1$ and $P2$, respectively. The interaction of populations in Algorithm 2 differs in selection from that in Algorithm 1. The reason is that when f = 1 in Algorithm 1, it means that UPF is the same as CPF, and the category is type-I [4]. When f is not equal to 1, the relationship between UPF and CPF is one of type-II: CPF is part of UPF, type-III: CPF and UPF partially overlap, and type-IV: CPF is wholly separated from UPF. Thus, in order to more efficiently select populations to enter the next generation of updates, we used a population success rate to judge.

2.3. Average Runtime and Computational Complexity

Table 2 shows the average running time of each algorithm on different test suites. We can see that the running time of CDCBM is not outstanding. The reason for this result is that all the operators are executed twice in one generation and CDCBM will consume more running time.

In this paper, the main complexity of the algorithm comes from the environment selection and evolutionary operators. The CDP method and the ε method are used for the main population and auxiliary population, respectively, and the complexity of both are O($M \cdot N^2$). In further, CDCBM uses the differential operator to generate offspring. The complexity of electing parents and generating offspring are O(N) and O($N \cdot D$), respectively. Thus, similar to most CMOEAs, the total computational complexity of CDCBM is 2·O(N) + 2·O($N \cdot D$) + 2·O($M \cdot N^2$) = O($M \cdot N^2$).

Table 2. Average running time of each algorithm on different test suites. Values highlighted in grey represent the best results achieved in each test question.

Problem	NSGAII	ToP	CMOEA_MS	CTAEA	CCMO	cDPEA	EMCMO	CDCBM
DASCMOP test suite	9.1011e+0	5.3855e+1	5.7323e+1	2.2914e+2	5.2263e+1	7.4306e+1	8.6454e+1	1.0713e+2
DOC test suite	8.452e+0	5.0376e+1	6.7325e+1	2.3801e+2	9.2740e+1	8.8582e+1	1.3091e+2	8.1245e+1
LIRCMOP test suite	1.066e+1	2.0223e+1	4.6690e+1	3.1756e+2	1.1176e+2	1.1719e+2	1.5505e+2	1.1420e+2
+/−/=	3/0/0	3/0/0	3/0/0	0/3/0	1/1/1	1/1/1	1/2/0	

3. Results

In Section 3.1, we describe the compared algorithms and parameter settings. Section 3.2 shows the results obtained by comparing our proposed CDCBM with other methods. In Section 3.3, we discuss the impact of changes in the constraint bound ε on the algorithm.

3.1. Experimental Settings

3.1.1. Compared Algorithms and Parameter Settings

Seven state-of-the-art algorithms are used to verify the performance of CDCBM. As Table 3 shows the classification of the comparison algorithms. Among these algorithms, NSGA-II [6], C-TAEA [8], cDPEA [18], and EMCMO [20] use simulated binary crossover [21] and polynomial mutation [22] to generate progeny solutions. CCMO [7] and CDCBM utilize differential evolution (DE) [23] to generate progeny solutions. To make a fair comparison, the population size N of all algorithms was set to 100, the number of evaluations FEs was set to 300,000, and the number of runs was set to 30. In the SBX and PM operators, the crossover probability and distribution index were set to 1 and 20, respectively, and the mutation probability and distribution index were set to 1/D and 20, respectively. All other parameters of the algorithm were the same as the original algorithm. All experiments in this paper were run on PlatEMO [24].

Table 3. Classification of comparison algorithms.

Methods Classification	Algorithm for Comparison
CDP method	NSGAII [6]
multi-stage method	ToP [14], CMOEA_MS [5]
multiple population method	CTAEA [8], CCMO [7], c-DPEA [18], EMCMO [20]

3.1.2. Test Functions and Performance Indicators

To verify the performance of CDCBM, we used three challenging benchmark suites for testing: DASCMOP [25], DOC [14] and LIRCMOP [26]. The decision vector D = 30 (D is the dimension of the decision variable in the decision space) for the DASCMOP [25] test suite defines the difficulty, diversity and convergence of feasibility. For the nine DOC [14] problems, D is fixed to different values according to different problems, and its dimension setting is the same as in [14]. Specifically, the main features of the LIRCMOP [26] test set are large infeasible regions and complex connections between location and distance variables. Moreover, this test suite consists of 14 functions with D = 10. The number of evaluations Fes for these three benchmark suites was set to 300,000.

In addition, two widely used performance metrics were employed: Inverted Intergenerational Distance (IGD) [27] and High Volume (HV) [28]. The smaller the required IGD value, the larger the HV value. The non-parametric Wilcoxon's rank-sum test [16] was performed on the IGD and HV results at the 95% confidence level [29]. The symbols "+", "−", and "=" indicate that the comparison algorithm is significantly better, worse than, or comparable to CDCBM, respectively. The best measure for each question is highlighted in grey in each table.

3.2. Comparisons with Other Methods

Table 4 shows the mean IGD values and standard deviations of eight algorithms on three test suites. As seen from the table, CDCBM achieved the best results on 21 problems, followed by EMCMO [20] with four best results. cDPEA [18], CCMO [7], CMOEA_MS [5], and ToP [14] also have 2, 1, 3, and 1 best results, respectively. Conversely, NSGA-II [6] and CTAEA [8] did not perform the best on any of the 32 problems. Table 1 shows that CDCBM is significantly better than NSGA-II [6], ToP [14], CMOEA_MS [5], CTAEA [8], CCMO [7], cDPEA [18], and EMCMO [20] on 28, 24, 22, 21, 25, 18, and 20 problems.

DASCMOP [25] functions: In this test suite, most feasible regions of DASCMOP [25] are disconnected and away from UPF. This makes the algorithm take into account diversity and convergence speed in order to obtain good results. As can be seen from the experimental results of the eight algorithms in Table 4, CDCBM achieved the best test results on the four problems DASCMOP1, DASCMOP2, DASCMOP3, and DASCMOP9. EMCMO [20] performed best on the two DASCMOP test questions. This is likely because most DASCMOP test questions have narrow and easily overlooked feasible regions. The dynamic constraint boundary method in our algorithm can effectively search these feasible regions, which other algorithms may have difficulty searching. Therefore, CDCBM is an algorithm that fits well with the DASCMOP function.

As can be seen from Figure 1, all algorithms except CDCBM converged to only part of the CPF. Because CDCBM employs a dynamic constrained boundary approach, after *P2* searches from the UPF to the approximate CPF, it will frequently search between the infeasible region and the CPF. The effective information provided by *P2* dramatically improves the distribution of P1.

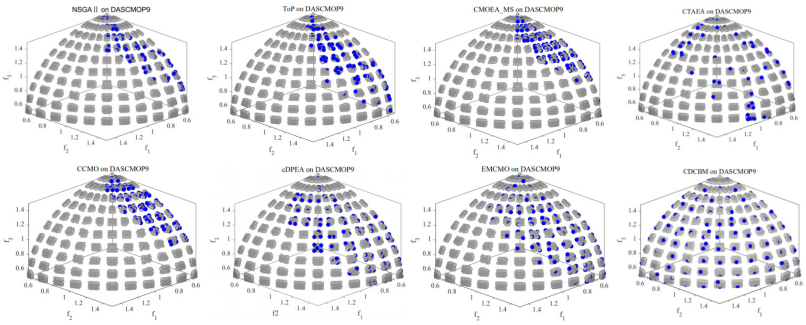

Figure 1. Solutions with median IGD value among 30 runs obtained by NSGA-II, ToP, CMOEA_MS, CTAEA, CCMO, cDPEA, EMCMO, and CDCBM on DASCMOP9.

DOC [14] functions: The DOC test suite is very challenging, in which nine test functions involve both decision space and target space constraints. According to its complex characteristics, the CPF can be continuous, discrete, mixed, or degenerate. As seen from the experimental results of the eight algorithms in Table 4, CDCBM achieved the seven best test results out of nine problems. Figure 2 reflects the performance of each algorithm. NSGA-II [6] is too concerned with satisfying constraints and thus fell into a local optimum, failing to converge to the true PF. Because ToP [14] did not fully utilize the feasible solutions found in the first stage, it did not converge. C-TAEA [8] cannot cross the infeasible barrier, so that no feasible solution can be found. Neither the auxiliary populations of cDPEA [18] nor EMCMO [20] could provide better CPF information resulting in poor performance. CDCBM outperformed DOC2 because the main population fully used the information of the promising infeasible solutions of the auxiliary population, resulting in better distribution. CCMO [7] converged to the CPF, but in terms of distribution, CDCBM is slightly better than CCMO [6].

LIRCMOP functions: Most of the test problems in LIRCMOP [26] problems contain many disjoint small feasible regions, hindered by extensive infeasible regions; the CPF usually consists of several disjoint segments or sparse points, and some CPFs even have just a curve. This poses severe challenges to maintaining population diversity. The experimental

results in Table 4 show that CDCBM performed the best on 10 problems. EMCMO [20], cDPEA [18], and CMOEA_MS [5] achieved the best results on 2, 1, and 1 problem, respectively, while the other four algorithms did not achieve the best results on this test suite.

Table 4. The IGD values obtained by CDCBM and seven compared algorithms on all three benchmark sets. The first figure is the mean IGD value, and the standard deviation is in parentheses. "NAN" means that the algorithm did not find any feasible solution in 30 runs. '+', '−', and '=' indicate better, worse, or equivalent results than CDCBM. Values highlighted in grey represent the best results achieved in each test question.

Problem	NSGAII	ToP	CMOEA_MS	CTAEA	CCMO	cDPEA	EMCMO	CDCBM
DASCMOP1	7.2526e-1 (4.20e-2) -	7.1495e-1 (1.61e-1) -	7.1430e-1 (3.60e-2) -	1.8267e-1 (1.42e-2) -	3.0908e-3 (3.51e-4) -	6.2863e-1 (1.18e-1) -	7.1207e-1 (2.50e-2) -	2.8942e-3 (2.64e-4)
DASCMOP2	2.6682e-1 (2.47e-2) -	3.3918e-1 (2.20e-1) -	2.4548e-1 (3.88e-2) -	8.0022e-2 (2.56e-2) -	4.3075e-3 (1.02e-4) -	2.1299e-1 (3.05e-2) -	2.3050e-1 (2.15e-2) -	4.1431e-3 (1.20e-4)
DASCMOP3	3.4928e-1 (4.61e-2) -	6.7959e-1 (1.42e-1) -	3.3637e-1 (3.54e-2) -	1.3292e-1 (2.59e-2) -	1.9490e-2 (1.38e-3) -	2.5961e-1 (1.80e-2) -	3.3073e-1 (5.29e-2) -	1.9423e-2 (1.07e-4)
DASCMOP4	1.5126e-3 (5.88e-5) +	NaN (NaN)	1.0540e-1 (1.21e-1) =	1.0054e-2 (1.97e-3) +	5.9074e-2 (2.32e-1) -	1.4042e-3 (8.13e-4) +	1.3630e-3 (7.27e-4) +	1.8328e-2 (6.80e-2)
DASCMOP5	3.5210e-3 (1.29e-4) -	NaN (NaN)	2.6366e-3 (2.60e-5) +	7.1873e-3 (5.01e-4) -	3.4477e-3 (4.07e-4) -	2.9818e-3 (1.81e-4) =	2.6994e-3 (4.29e-5) +	3.0525e-3 (1.76e-4)
DASCMOP6	2.7740e-1 (1.52e-1) -	NaN (NaN)	6.0948e-2 (1.09e-1) =	2.3439e-2 (3.31e-3) +	2.4549e-2 (2.24e-2) =	1.8585e-2 (6.01e-3) =	1.9668e-2 (4.27e-3) =	8.5347e-2 (1.87e-1)
DASCMOP7	4.9706e-2 (2.96e-3) +	NaN (NaN)	3.1097e-2 (6.03e-4) +	3.8368e-2 (6.59e-4) =	5.3327e-2 (6.12e-2) +	3.1521e-2 (7.08e-4) +	3.1092e-2 (7.12e-4) +	6.5238e-2 (1.41e-1)
DASCMOP8	6.2230e-2 (3.41e-3) -	NaN (NaN)	4.0033e-2 (8.71e-4) +	5.4630e-2 (1.38e-3) -	5.5127e-2 (4.47e-2) =	4.0323e-2 (1.27e-3) +	4.0637e-2 (1.02e-3) +	5.3820e-2 (4.58e-2)
DASCMOP9	3.6499e-1 (7.05e-2) -	5.1037e-1 (2.54e-1) -	3.3171e-1 (8.71e-2) -	1.9285e-1 (5.72e-2) -	4.1004e-2 (1.25e-3) =	1.7576e-1 (7.98e-2) -	3.3973e-1 (7.26e-2) -	4.0855e-2 (7.82e-4)
DOC1	2.1991e+0 (2.23e+0) -	5.8883e-3 (2.02e-4) -	2.5919e+0 (2.09e+0) -	4.8136e+2 (2.03e+2) -	5.8017e-3 (5.23e-4) -	3.9834e-1 (4.38e-1) -	3.3677e+0 (2.62e+0) -	5.0776e-3 (2.64e-4)
DOC2	NaN (NaN)	4.8965e-1 (6.34e-2) -	NaN (NaN)	NaN (NaN)	3.5842e-2 (1.10e-1) -	NaN (NaN)	NaN (NaN)	3.1271e-3 (4.85e-4)
DOC3	6.9740e+2 (1.76e+2) -	5.8651e+1 (9.27e+1) =	6.7349e+2 (2.48e+2) -	NaN (NaN)	4.8303e+2 (4.00e+2) -	7.9037e+2 (2.22e+2) -	6.8813e+2 (2.18e+2) -	1.1738e+2 (1.95e+2)
DOC4	7.1454e-1 (4.89e-1) -	5.3342e-2 (4.23e-2) =	6.6783e-1 (5.53e-1) -	2.3145e+2 (2.78e+2) -	2.3633e-2 (4.98e-3) -	4.5133e-1 (1.72e-1) -	1.2236e+0 (1.45e+0) -	1.6543e-2 (3.00e-3)
DOC5	NaN (NaN)	4.1792e+1 (5.33e+1) =	9.1339e+1 (2.65e+1) -	NaN (NaN)	6.9211e+0 (2.99e+1) =	NaN (NaN)	NaN (NaN)	4.3559e+1 (6.26e+1)
DOC6	2.5644e+0 (2.60e+0) -	2.2025e+0 (9.64e-1) -	2.2944e+0 (2.06e+0) -	7.3871e+1 (1.97e+2) -	4.7150e-3 (2.41e-3) -	2.8721e+0 (3.64e+0) -	2.0132e+0 (1.86e+0) -	2.5439e-3 (1.08e-4)
DOC7	5.9148e+0 (2.01e+0) -	2.8786e-1 (2.21e-1) -	4.4864e+0 (2.24e+0) -	NaN (NaN)	2.5442e-3 (3.14e-4) -	8.5322e+0 (2.61e+0) -	5.8704e+0 (2.60e+0) -	2.4047e-3 (2.27e-4)
DOC8	5.7676e+1 (4.81e+1) -	1.6934e+1 (1.44e+1) -	1.5135e+2 (7.28e+1) -	4.4807e+2 (1.07e+2) -	7.2907e-2 (4.68e-3) =	6.8493e+1 (5.51e+1) -	5.7484e+1 (6.15e+1) -	7.1668e-2 (4.31e-3)
DOC9	1.9870e-1 (8.70e-2) -	1.8463e-1 (3.60e-2) -	7.7955e-2 (9.39e-2) -	7.7038e-1 (2.58e-1) -	8.1110e-2 (1.04e-2) -	1.5170e-1 (1.18e-1) =	1.8271e-1 (1.10e-1) -	6.9206e-2 (7.80e-3)
LIRCMOP1	2.0084e-1 (6.17e-2) -	1.0620e-1 (9.08e-2) -	3.5832e-1 (1.55e-1) -	1.7586e-1 (1.06e-1) -	3.4921e-2 (2.28e-2) -	1.2016e-1 (1.04e-1) -	1.0701e-1 (7.04e-2) -	1.1219e-2 (5.12e-3)
LIRCMOP2	1.7298e-1 (6.90e-2) -	9.5894e-2 (1.00e-1) -	2.7243e-1 (1.00e-1) -	5.9282e-2 (1.49e-2) -	9.1456e-2 (4.27e-2) -	7.5270e-2 (5.60e-2) -	4.6155e-2 (2.10e-2) -	1.3184e-2 (1.32e-2)
LIRCMOP3	2.3397e-1 (8.54e-2) -	3.2542e-1 (6.86e-2) -	3.9437e-1 (1.25e-1) -	1.9567e-1 (1.30e-1) -	1.5096e-1 (5.01e-2) -	1.1532e-1 (8.41e-2) -	1.2843e-1 (6.24e-2) -	3.2199e-2 (1.95e-2)
LIRCMOP4	2.1612e-1 (5.55e-2) -	3.2224e-1 (4.63e-2) -	2.8876e-1 (9.68e-2) -	1.2122e-1 (4.54e-2) -	1.3924e-1 (5.20e-2) -	1.0630e-1 (5.68e-2) -	1.1990e-1 (4.66e-2) -	3.0364e-2 (1.99e-2)
LIRCMOP5	5.8592e-1 (5.40e-1) -	1.3819e-1 (3.55e-1) -	1.0349e-2 (7.93e-3) -	6.5429e-2 (2.09e-2) -	5.9397e-3 (2.22e-4) -	6.8125e-3 (1.31e-3) -	5.9978e-3 (7.99e-4) -	4.2949e-3 (1.33e-4)
LIRCMOP6	4.5287e-1 (5.14e-1) -	1.7174e-2 (5.83e-2) -	1.3030e-2 (2.75e-2) -	9.1932e-2 (9.37e-2) -	5.6770e-3 (2.31e-4) -	6.2217e-3 (2.28e-4) -	5.6705e-3 (2.36e-4) -	5.0090e-3 (3.29e-4)
LIRCMOP7	9.3994e-3 (1.23e-3) -	8.5713e-3 (3.17e-4) -	1.0925e-2 (1.53e-2) =	1.7431e-2 (2.66e-3) -	7.1964e-3 (2.39e-4) =	7.1121e-3 (2.86e-4) =	7.2658e-3 (3.20e-4) -	7.0960e-3 (2.45e-4)
LIRCMOP8	2.8821e-2 (5.88e-2) -	8.6180e-3 (3.43e-4) -	7.0103e-3 (1.40e-3) +	1.7384e-2 (3.82e-3) -	7.0626e-3 (2.69e-4) =	6.9505e-3 (1.79e-4) +	7.0566e-3 (1.87e-4) =	7.0525e-3 (2.20e-4)
LIRCMOP9	4.4821e-1 (1.09e-1) -	2.4918e-1 (1.71e-1) -	2.4273e-1 (1.38e-1) -	5.4475e-2 (2.96e-2) -	3.8872e-3 (1.41e-3) -	3.3889e-2 (3.66e-2) -	3.3783e-3 (1.11e-3) -	2.3225e-3 (4.72e-5)
LIRCMOP10	2.8280e-1 (9.05e-2) -	5.4624e-3 (2.52e-4) -	8.0527e-2 (4.74e-2) -	7.4161e-2 (7.88e-2) -	4.4496e-3 (1.84e-4) -	4.5673e-3 (1.60e-4) -	4.5035e-3 (2.19e-4) -	3.7414e-3 (9.22e-5)
LIRCMOP11	1.5103e-1 (1.18e-1) -	1.1039e-1 (7.55e-2) -	7.0821e-2 (3.46e-2) -	1.1945e-1 (3.30e-2) -	2.3690e-3 (4.38e-5) =	2.3836e-3 (4.26e-5) =	2.3644e-3 (4.69e-5) =	2.3821e-3 (4.83e-5)
LIRCMOP12	1.0246e-1 (5.01e-2) -	3.4008e-2 (5.43e-2) -	4.9910e-2 (6.68e-2) -	1.6254e-2 (5.13e-3) -	3.0268e-3 (8.46e-5) -	2.9980e-3 (6.40e-5) =	2.9927e-3 (6.35e-5) =	2.9626e-3 (8.38e-5)
LIRCMOP13	1.2032e-1 (4.63e-3) -	1.2569e-1 (3.83e-3) -	9.2337e-2 (8.75e-4) +	1.0859e-1 (2.20e-3) -	9.3242e-2 (7.70e-4) +	9.3090e-2 (1.07e-3) +	9.0581e-2 (5.68e-4) +	1.0507e-1 (2.70e-3)
LIRCMOP14	1.6081e-1 (2.11e-1) -	1.1886e-1 (3.86e-3) -	9.4946e-2 (8.04e-4) +	1.1139e-1 (8.78e-4) -	9.5920e-2 (9.27e-4) +	9.5591e-2 (8.51e-4) +	9.5534e-2 (8.98e-4) +	9.9546e-2 (1.19e-3)
+/-/=	2/28/0	0/24/3	6/22/3	2/25/1	3/21/8	6/18/6	6/20/4	

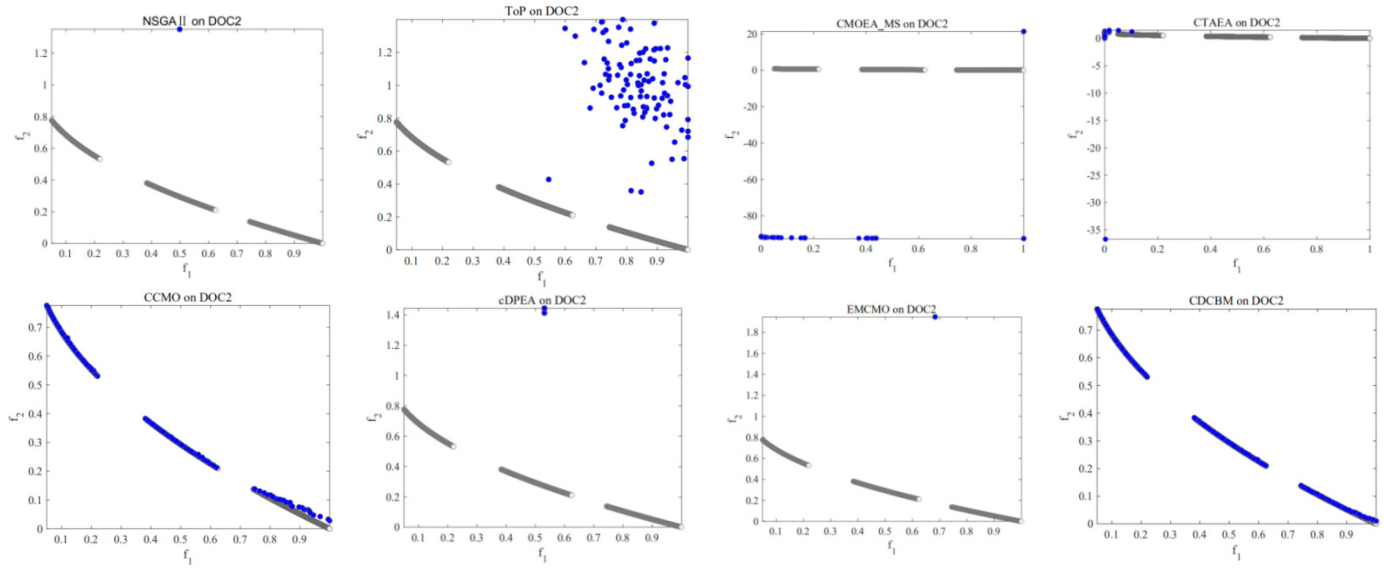

Figure 2. Solutions with median IGD value among 30 runs obtained by NSGA-II, ToP, CMOEA_MS, CTAEA, CCMO, cDPEA, EMCMO, and CDCBM on DOC2.

As shown in Figure 3, since the feasible region of LIRCMOP4 is only a discontinuous line segment, NSGA-II [6], ToP [14], CTAEA [8], CCMO [7], cDPEA [18], and EMCMO [20] all fall into local optima. However, only CDCBM found all discontinuous CPFs.

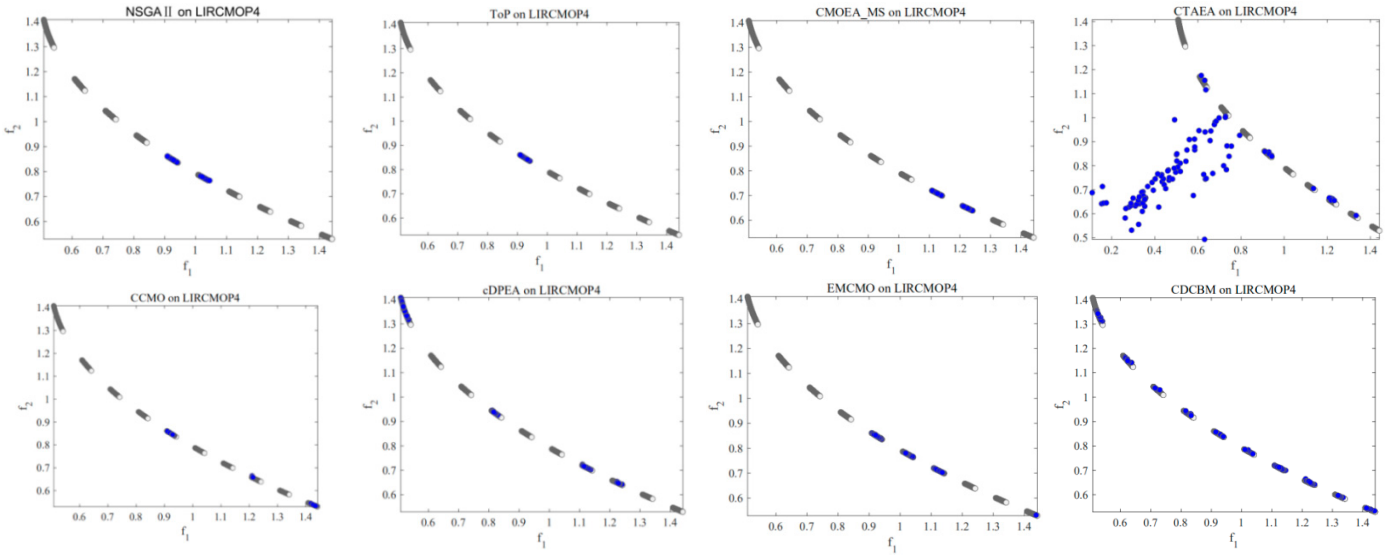

Figure 3. Solutions with median IGD value among 30 runs obtained by NSGA-II, ToP, CMOEA_MS, CTAEA, CCMO, cDPEA, EMCMO, and CDCBM on LIRCMOP4.

Real-world CMOPs: The comparison of our algorithm with the comparative algorithm on three real-world problems is presented here. The Spring Design problem [30] has two optimization objectives, eight inequality constraints, and three decision variables. The synchronous optimal pulse width modulation of 3-level and 5-level inverter problem [31,32] has two optimization objectives, 24 inequality constraints, and 25 decision variables. The definitions of all optimization objectives and constraints can be found in their references. The HV results of our algorithm and the comparative algorithm are given in Table 5. From Table 5, we can see that our algorithm obtains the maximum HV on these three real-world problems, indicating that our algorithm achieves the best performance on these three problems.

Table 5. The HV values obtained by CDCBM and seven compared algorithms on all three benchmark sets. The first figure is the mean HV values, and the standard deviation is in parentheses. "NAN" means that the algorithm did not find any feasible solution in 30 runs. '+', '−', and '=' indicate better, worse, or equivalent results than CDCBM. Values highlighted in grey represent the best results achieved in each test question.

Problem	NSGAII	ToP	CMOEA_MS	CTAEA	CCMO	cDPEA	EMCMO	CDCBM
Spring Design (3/2/8)	5.4360e-1 (3.36e-5) -	5.4370e-1 (7.25e-6) =	5.4370e-1 (5.10e-5) -	5.4273e-1 (2.26e-4) -	5.4366e-1 (9.80e-6) -	5.4348e-1 (1.89e-4) -	5.4357e-1 (1.74e-4) -	5.4370e-1 (8.64e-6)
Synchronous Optimal Pulse-width Modulation of 3-level Inverters (25/2/24)	5.9793e-1 (1.33e-1) -	5.4396e-1 (1.29e-1) -	5.2816e-1 (0.00e+0) =	4.0345e-1 (3.69e-1) -	6.8501e-1 (1.36e-1) =	6.0888e-1 (8.06e-2) -	6.0240e-1 (1.36e-1) -	7.5613e-1 (7.39e-2)
Synchronous Optimal Pulse-width Modulation of 5-level Inverters (25/2/24)	3.9657e-1 (3.18e-1) -	2.1521e-1 (2.56e-1) -	5.9256e-2 (0.00e+0) =	3.2158e-1 (0.00e+0) =	5.8045e-1 (2.70e-1) =	4.6157e-1 (3.12e-1) -	4.2138e-1 (3.42e-1) -	7.4714e-1 (1.65e-1)
+/−/=	0/3/0	0/2/1	0/1/2	0/2/1	0/1/2	0/3/0	0/3/0	

3.3. Further Investigations of CDCBM

In this subsection, we discuss the impact of changes in the constraint bound ε on the algorithm, which compares CDCBM with two variants on the LIRCMOP benchmark suite. The first variant of CDCBM adopts a decreasing bound, and the second variant bounds value to infinity all the time during the evolution. These two variables verify that the dynamic constraint bounds are valid.

Table 6 presents the performance of CDCBM and its two variants on the LIRCMOP suite. We can see that CDCBM gets the eight best averages on LIRCMOP. Although Variant 1 and Variant 2 obtained five and one best averages, respectively, they did not show significant differences. Furthermore, CDCBM has five and seven results that significantly outperform Variant 1 and Variant 2, respectively, demonstrating the effectiveness of dynamically constrained boundaries.

Table 6. The IGD values obtained by CDCBM and its two variants on the DASCMOP benchmark suite. The first figure is the mean IGD values, and the standard deviation is in parentheses. Best result in each row is highlighted.

Problem	Variant1_CDCBM	Variant2_CDCBM	CDCBM
LIRCMOP1	1.8778e-2 (8.64e-3) -	4.0770e-2 (2.98e-2) -	1.1219e-2 (5.12e-3)
LIRCMOP2	3.4966e-2 (2.69e-2) -	7.9022e-2 (3.98e-2) -	1.3184e-2 (1.32e-2)
LIRCMOP3	7.2556e-2 (3.83e-2) -	1.3058e-1 (4.21e-2) -	3.2199e-2 (1.95e-2)
LIRCMOP4	6.8050e-2 (3.61e-2) -	1.2901e-1 (3.80e-2) -	3.0364e-2 (1.99e-2)
LIRCMOP5	4.2601e-3 (1.51e-4) =	4.2963e-3 (1.19e-4) =	4.2949e-3 (1.33e-4)
LIRCMOP6	5.1505e-3 (2.58e-4) =	4.9998e-3 (3.96e-4) =	5.0090e-3 (3.29e-4)
LIRCMOP7	7.0018e-3 (2.02e-4) =	7.1588e-3 (1.63e-4) =	7.0960e-3 (2.45e-4)
LIRCMOP8	7.0198e-3 (2.21e-4) =	7.2061e-3 (2.17e-4) -	7.0525e-3 (2.20e-4)
LIRCMOP9	2.3166e-3 (4.60e-5) =	2.6348e-3 (6.78e-5) -	2.3225e-3 (4.72e-5)
LIRCMOP10	3.7228e-3 (1.09e-4) =	4.3685e-3 (1.24e-4) -	3.7414e-3 (9.22e-5)
LIRCMOP11	2.3889e-3 (3.73e-5) =	2.3848e-3 (4.34e-5) =	2.3821e-3 (4.83e-5)
LIRCMOP12	3.0224e-3 (8.57e-5) -	3.0235e-3 (1.26e-4) =	2.9626e-3 (8.38e-5)
LIRCMOP13	1.0546e-1 (2.09e-3) =	1.0439e-1 (2.72e-3) =	1.0507e-1 (2.70e-3)
LIRCMOP14	9.9801e-2 (1.27e-3) =	9.9705e-2 (9.10e-4) =	9.9546e-2 (1.19e-3)
+/−/=	0/5/9	0/7/7	

4. Discussion

In this section, to demonstrate the effectiveness of our algorithm, we visualize the running process of seven algorithm, NSGA-II [6], ToP [14], CMOEA_MS [5], CTAEA [8], CCMO [7], cDPEA [18], EMCMO [20], and CDCBM. Figure 4 presents the population distribution of all seven CMOEAs on LIRCMOP3 at different generations.

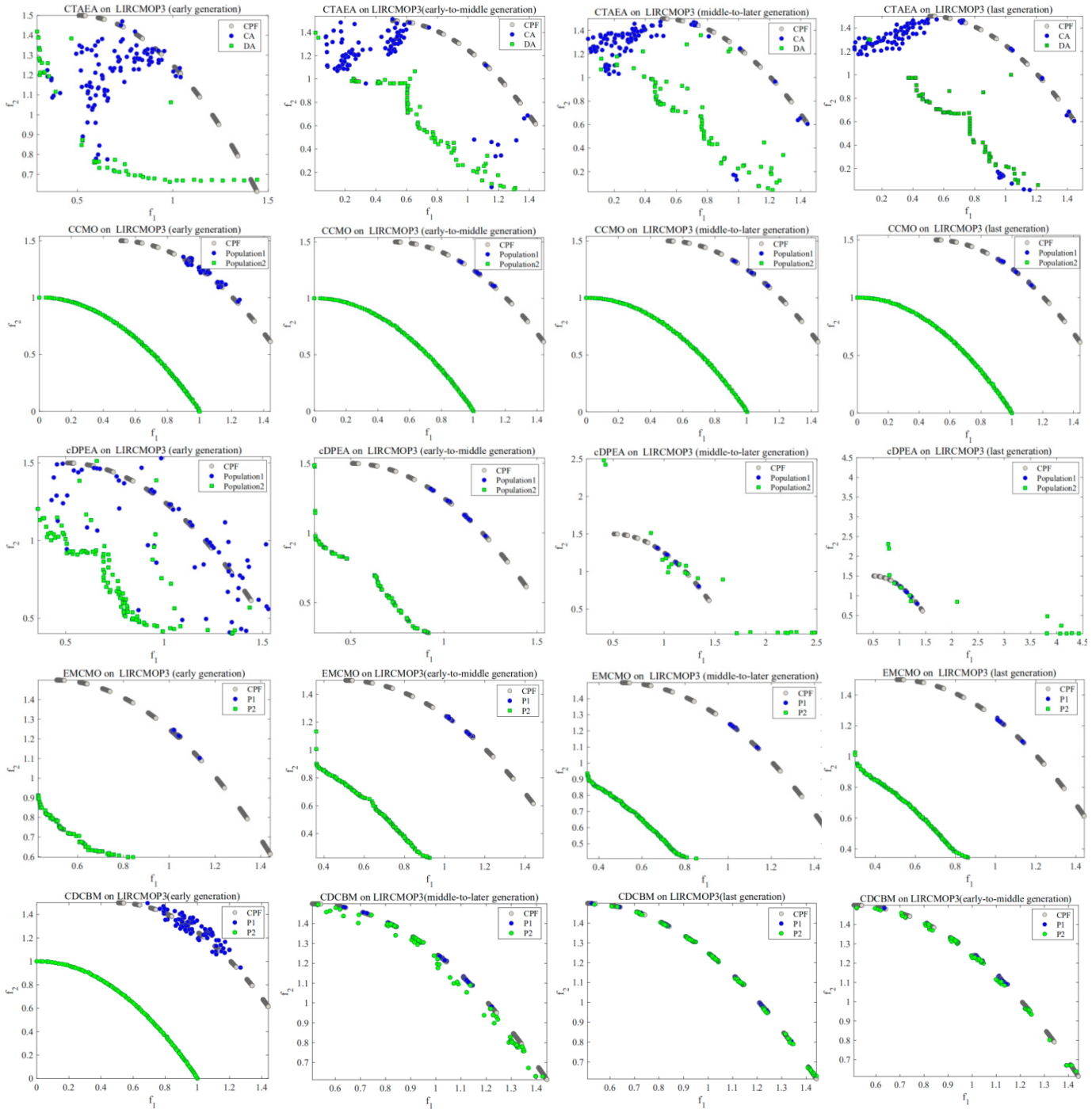

Figure 4. Population distribution of five CMOEAs on LIRCMOP3 in different generations, where the grey line is CPF, and infeasible regions are represented by the white regions.

CTAEA: CTAEA [8] includes a Convergence Oriented Archive (CA) and a Diversity Oriented (DA). Most of the parents in this algorithm are from CA and DA, and DA does not consider any constraints to provide CA with information about feasible regions that

it has not explored. Early CA considers both objective and constraint and can partially converge to the feasible region, but most infeasible solutions will still exist in the population. However, the limited mate selection scheme in the middle and late stages affected the evolutionary direction of CA, and most of the offspring were located between CPF and UPF. Therefore, the convergence and diversity of CA are relatively poor.

CCMO: Population 1 considers constraints in the early stages of evolution and quickly converges to the local CPF, while population 2 does not consider constraints at all and converges to UPF. In the middle and late stages, CCMO [7] can obtain a well-distributed population 2. However, population 2 cannot continuously provide effective information for population 1 and cannot help population 1 to jump out of the local optimum, resulting in poor distribution.

cDPEA: cDPEA has better distribution in the early stage, thanks to adopting a self-adaptive penalty function. Then, the main population converges to the local CPF, and the auxiliary population converges to the UPF. In the middle-to-later stage, the auxiliary population starts to search between CPF and UPF, and some individuals come to the vicinity of the feasible region. However, the diversity of the auxiliary population is too poor to help the main population effectively. Finally, the main population is still limited to part of the CPF.

EMCMO: In this algorithm, CMOP is modeled as a multi-task optimization problem, the first task considers both constraints and objectives (i.e., CPF), and the auxiliary task is to find a well-distributed UPF. Its early stage is the same as CCMO [7]. In the early-to-middle and middle-to-later generations, the second task can better converge to the UPF. Furthermore, the designed heuristic method finds valuable knowledge and carries out knowledge transfer. Nevertheless, the CPF of LIRCMOP3 is far away from the UPF, which reduces the effect of knowledge transfer. This also makes the distribution of the main tasks between the middle and late generation and the previous generation the same, and neither can jump out of the local optimal area.

CDCBM: In the early stage, *P2* converges to the UPF regardless of constraints, and the generated descendants help *P1* converge to the feasible region. In the early-to-middle generation, as the ε gradually decreases, *P2* gradually comes to the boundary of the feasible region. In the process of searching from UPF to CPF, *P2* retains many potential infeasible solutions. These solutions help P1 find most of the CPFs, but some CPFs are still ignored during evolution. Therefore, in the middle-to-later generation, ε increases to a (a constraint tolerance value that gradually decreases with the evolutionary algebra), and *P2* returns to the feasible region boundary to repeat the search. In this process, we will also judge the success rate of the parent and the offspring to more effectively provide a complementary evolutionary direction for *P1*. In the final generation, *P1* finds all feasible regions and has a relatively good distribution.

5. Conclusions

We have proposed a MOEA based on the dynamic constraint boundary method. The auxiliary population evolves to the UPF without initially considering the constraints, and as the constraint boundary decreases, the population moves closer to the feasible region. These potentially infeasible solutions can help the main population to overcome obstacles in the infeasible region. In addition, when the auxiliary population reaches the CPF, the constraint boundary changes again, and the population returns to search near the feasible region. This operation preserves some well-distributed infeasible solutions to help the main population find potential unsearched feasible regions. Moreover, in the evolution process, according to the state of the auxiliary population, the effective parent and child individuals will be selected to enter the next generation update, which can effectively reduce the waste of iteration times.

Although the proposed CDCBM performed well on these several test suites, several aspects must be investigated. If all initial solutions are feasible, ε always takes infinity. In this case, it is necessary to find a way to set an appropriate initial value because it has a

relatively significant impact on the evolution of the auxiliary population. In addition, we will explore our algorithm parameters in more detail to better improve our algorithm.

Author Contributions: Conceptualization, Z.L.; methodology, Q.W.; software, Q.W.; validation, Z.L.; formal analysis, Z.L.; investigation, Z.L.; resources, J.Z.; writing—original draft preparation, Z.L.; writing—review and editing, Z.L.; visualization, Z.L.; supervision, X.Y.; project administration, Y.L., Y.H. and Y.X.; funding acquisition, Q.W. All authors have read and agreed to the published version of the manuscript.

Funding: The authors wish to thank the support of the National Natural Science Foundation of China (Grant No. 61876164), the Natural Science Foundation of Hunan Province (Grant No. 2022JJ40452), Doctoral research start-up project (Grant No. 22QDZ03).

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to the data provider's request for confidentiality of the data and results.

Acknowledgments: We thank our institute teachers and students for their support in the work of this paper.

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

References

1. Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*; John Wiley & Sons: Chichester, UK, 2001.
2. Wang, J.; Ren, W.; Zhang, Z.; Huang, H.; Zhou, Y. A hybrid multiobjective memetic algorithm for multiobjective periodic vehicle routing problem with time windows. *IEEE Trans. Syst. Man. Cybern. Syst.* **2020**, *50*, 4732–4745. [CrossRef]
3. Tanabe, R.; Oyama, A. A note on constrained multi-objective optimization benchmark problems. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 1127–1134.
4. Ma, Z.; Wang, Y. Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons. *IEEE Trans. Evol. Comput.* **2019**, *23*, 972–986. [CrossRef]
5. Tian, Y.; Zhang, Y.; Su, Y.; Zhang, X.; Tan, K.C.; Jin, Y. Balancing objective optimization and constraint satisfaction in constrained evolutionary multiobjective optimization. *IEEE Trans. Cybern.* **2021**. [CrossRef] [PubMed]
6. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.A.M.T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
7. Tian, Y.; Zhang, T.; Xiao, J.; Zhang, X.; Jin, Y. A coevolutionary framework for constrained multiobjective optimization problems. *IEEE Trans. Evol. Comput.* **2020**, *25*, 102–116. [CrossRef]
8. Li, K.; Chen, R.; Fu, G.; Yao, X. Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Trans. Evol. Comput.* **2018**, *23*, 303–315. [CrossRef]
9. Liang, J.; Ban, X.; Yu, K.; Qu, B.; Qiao, K.; Yue, C.; Chen, K.; Tan, K.C. A Survey on Evolutionary Constrained Multi-objective Optimization. *IEEE Trans. Evol. Comput.* **2022**. [CrossRef]
10. Ma, Z.; Wang, Y.; Song, W. A new fitness function with two rankings for evolutionary constrained multiobjective optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *51*, 5005–5016. [CrossRef]
11. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
12. Fan, Z.; Li, W.; Cai, X.; Hu, K.; Lin, H.; Li, H. Angle-based constrained dominance principle in MOEA/D for constrained multi-objective optimization problems. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 460–467.
13. Fan, Z.; Li, W.; Cai, X.; Li, H.; Wei, C.; Zhang, Q.; Deb, K.; Goodman, E. Push and pull search for solving constrained multi-objective optimization problems. *Swarm Evol. Comput.* **2019**, *44*, 665–679. [CrossRef]
14. 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]
15. Ma, H.; Wei, H.; Tian, Y.; Cheng, R.; Zhang, X. A multi-stage evolutionary algorithm for multi-objective optimization with complex constraints. *Inf. Sci.* **2021**, *560*, 68–91. [CrossRef]
16. Zhu, Q.; Zhang, Q.; Lin, Q. A constrained multiobjective evolutionary algorithm with detect-and-escape strategy. *IEEE Trans. Evol. Comput.* **2020**, *24*, 938–947. [CrossRef]
17. 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**, *26*, 1129–1143. [CrossRef]
18. Ming, M.; Trivedi, A.; Wang, R.; Srinivasan, D.; Zhang, T. A dual-population-based evolutionary algorithm for constrained multiobjective optimization. *IEEE Trans. Evol. Comput.* **2021**, *25*, 739–753. [CrossRef]
19. Zou, J.; Sun, R.; Yang, S.; Zheng, J. A dual-population algorithm based on alternative evolution and degeneration for solving constrained multi-objective optimization problems. *Inf. Sci.* **2021**, *579*, 89–102. [CrossRef]

20. Qiao, K.; Yu, K.; Qu, B.; Liang, J.; Song, H.; Yue, C. An evolutionary multitasking optimization framework for constrained multiobjective optimization problems. *IEEE Trans. Evol. Comput.* **2022**, *26*, 263–277. [CrossRef]
21. Deb, K.; Agrawal, R.B. Simulated binary crossover for continuous search space. *Complex Syst.* **1995**, *9*, 115–148.
22. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338. [CrossRef]
23. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2010**, *15*, 4–31. [CrossRef]
24. Tian, Y.; Cheng, R.; Zhang, X.; Jin, Y. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum. *IEEE Comput. Intell. Mag.* **2017**, *12*, 73–87. [CrossRef]
25. Fan, Z.; Li, W.; Cai, X.; Li, H.; Wei, C.; Zhang, Q.; Deb, K.; Goodman, E. Difficulty adjustable and scalable constrained multiobjective test problem toolkit. *Evol. Comput.* **2020**, *28*, 339–378. [CrossRef] [PubMed]
26. Fan, Z.; Li, W.; Cai, X.; Huang, H.; Fang, Y.; You, Y.; Mo, J.; Wei, C.; Goodman, E. An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions. *Soft Comput.* **2019**, *23*, 12491–12510. [CrossRef]
27. 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]
28. While, L.; Hingston, P.; Barone, L.; Huband, S. A faster algorithm for calculating hypervolume. *IEEE Trans. Evol. Comput.* **2006**, *10*, 29–38. [CrossRef]
29. Zitzler, E.; Knowles, J.; Thiele, L. Quality assessment of pareto set approximations. *Multiobject. Optim.* **2008**, *52*, 373–404.
30. Kannan, B.; Kramer, S.N. An augmented lagrange multiplier based method for mixed integer iscrete continuous optimization and its applications to mechanical design. *J. Mech. Des.* **1994**, *116*, 405–411. [CrossRef]
31. Rathore, A.; Holtz, J.; Boller, T. Optimal pulsewidth modulation of multilevel inverters for low switching frequency control of medium voltage high power industrial ac drives. In Proceedings of the 2010 IEEE Energy Conversion Congress and Exposition, Atlanta, GA, USA, 12–16 September 2010. [CrossRef]
32. Rathore, A.; Holtz, J.; Boller, T. Synchronous optimal pulsewidth modulation for low-switching-frequency control of medium-voltage multilevel inverters. *Ind. Electron. IEEE Trans.* **2010**, *57*, 2374–2381. [CrossRef]