mathematics

MDPI

Article

Population Feasibility State Guided Autonomous Constrained Multi-Objective Evolutionary Optimization

Mingcheng Zuo * and Yuan Xue

Artificial Intelligence Research Institute, China University of Mining and Technology, Xuzhou 221116, China; ts23810019p31@cumt.edu.cn

* Correspondence: mingcheng.zuo@cumt.edu.cn; Tel.: +86-13349987465

Abstract: Many practical problems can be classified as constrained multi-objective optimization problems. Although various methods have been proposed for solving constrained multi-objective optimization problems, there is still a lack of research considering the integration of multiple constraint handling techniques. Given this, this paper combines the objective and constraint separation method with the multi-operator method, proposing a population feasibility state guided autonomous constrained evolutionary optimization method. This method first defines the feasibility state of the population based on both feasibility and ε feasibility of the solutions. Subsequently, a reinforcement learning model is employed to construct a mapping model between the population state and reproduction operators. Finally, based on the real-time population state, the mapping model is utilized to recommend the promising reproduction operator for the next generation. This approach demonstrates significant performance improvement for ε constrained mechanisms in constrained multi-objective optimization algorithms, and shows considerable advantages in comparison with state-of-the-art constrained multi-objective optimization algorithms.

Keywords: constrained multi-objective optimization problems; population feasibility state; autonomy; evolutionary optimization; reinforcement learning

MSC: 49M41

Citation: Zuo, M.; Xue, Y. Population Feasibility State Guided Autonomous Constrained Multi-Objective Evolutionary Optimization. *Mathematics* **2024**, *12*, 913. https://doi.org/10.3390/math12060913

Academic Editors: Andrea Scozzari and Ioannis G. Tsoulos

Received: 28 January 2024
Revised: 11 March 2024
Accepted: 13 March 2024
Published: 20 March 2024

1. Introduction

Constrained multi-objective optimization problems (CMOPs) refer to optimizing multiple conflicting performance metrics simultaneously while ensuring that decision variables meet the constraints of the objective problem. Many real-world problems can be classified as CMOPs, such as integrated energy dispatch [1], multi-stage portfolio [2], and spacecraft orbit optimization [3]. Without loss of generality, a CMOP can be defined as

$$\begin{aligned} \text{Min} \quad & F(x) = (f_1(x), f_2(x), \ldots, f_m(x)) \\ \text{s.t.} \quad & g_i(x) \le 0,\ i = 1, 2, \ldots, l \\ & h_j(x) = 0,\ j = l+1, l+2, \ldots, n \\ & x = (x_1, x_2, \ldots, x_d, \ldots, x_D),\ x_d \in rand(LB_d, UB_d) \end{aligned}$$

where x represents the $D-dimensional$ decision variables, $f_k(x)$ is the k-th objective function for $k = 1, 2, \ldots, m$. $g_i(x) \le 0$ represents the i-th inequality constraint for $i = 1, 2, \ldots, l$. $h_j(x) = 0$ represents the j-th equality constraint for $j = l+1, l+2, \ldots, n$. LB_d and UB_d are lower and upper boundaries of x_d, respectively, and $rand(LB_d, UB_d)$ generates a random number between LB_d and UB_d. The wide application in real-world problems makes the performance testing of the constrained multi-objective optimization algorithm more demanding; therefore, many real-world test suites, such as RC [4], have been proposed in the existing research, and the challenges of solving constrained multi-objective optimization problems can be studied in depth.

While meeting the problem constraints, the decision variables in CMOPs also need to be optimized for improving multiple conflicting objective functions simultaneously. This difficulty contributes to the challenging nature of solving multi-objective optimization problems. The key to solving CMOPs lies in constraint handling. Applying appropriate constraint handling methods ensures a balance between the feasibility of solutions, convergence, and diversity in the objective space. Currently, common constraint handling methods include the penalty function method, the objective and constraint separation method, the multi-objective method, the transformation method, the mixed method, and the multiple-operator method [5]. In the objective and constraint separation method, the ε constraint method is an effective constraint handling approach, which is capable of leveraging information from relatively good infeasible solutions to guide the evolution of the population to some extent. However, the ε constraint method also has limitations. For example, the setting of ε values often relies on human experience, which can lead to biases in the population search direction. It is, therefore, important to adjust the search direction for the ε constraint method. To adjust the search direction, the multiple-operator method is a candidate. In this method, use of different reproduction operators leads to the generation of diverse types of solutions, aiding in adjusting the search direction of the population [3,6–8]. However, the selection of reproduction operators also usually relies on subjective human experience, making it tough to determine the timing and scope of their application. Therefore, combining the ε constraint method with the multi-operator method poses a challenge.

In light of this, this paper combines the objective and constraint separation method with the multi-operator method, proposing a population feasibility state guided autonomous constrained evolutionary optimization method. This method first establishes the population state based on both feasibility and ε feasibility of the solutions. Subsequently, a reinforcement learning model is employed to construct a mapping model between the population state and the operator. Finally, based on the real-time population state, the mapping model is utilized to recommend the subsequent reproduction operator for the next generation.

The main contributions of this paper are summarized as follows:

(1) Proposing a self-guided optimization method that combines the objective and constraint separation method with the multi-operator method. Specifically, this method combines the ε constraint method with the multi-operator method, using the multi-operator approach to enhance the diversity of solutions with more search directions. Adjusting the population search direction helps reduce the dependence of the ε constraint method on human experience for setting ε values. Additionally, overcoming the limitations of the multi-operator method relying on human experience using reinforcement learning further enhances the algorithm's performance. This holds potential to contribute to the field of combining various constraint handling methods.

(2) Proposing a novel method for characterizing the population state. Existing research has not characterized the population state from the perspective of both feasibility and ε feasibility of population solutions, limiting the perception of the population's feasibility state. The proposed method for characterizing population state in this paper contributes to the description of population state in the field of autonomous intelligent optimization.

The rest of this paper is organized as follows. Section 2 reviews the relevant research works and points out the existing problems in the current research. The proposed method is described in Section 3, including the characterization of the population state, the population's reproduction operators, and their performance evaluation. Section 4 is the experimental results and analysis. Finally, Section 5 summarizes the whole paper and discusses future research directions.

2. Related Works

This paper proposes a solving approach for constrained multi-objective optimization problems by combining the objective and constraint separation method with the multi-operator method. Hence, this section primarily reviews existing methods for constraint handling. Given that our method focuses on the objective and constraint separation method, as well as the multi-operator method, a review of the progress of research on these two approaches is also provided.

2.1. Constrained Multi-Objective Evolutionary Optimization

Constraint handling is a crucial task in addressing constrained multi-objective problems. The penalty function method is a common approach for constraint handling. The idea is to construct a penalty term based on the degree of constraint violation and incorporate this penalty term into the objective function. This transforms the constrained optimization problem into an unconstrained optimization problem. The setting of the penalty factor has a significant impact on the algorithm. Setting the penalty factor to be too large or too small both have negative effects. If the feasible region consists of several disconnected and independent subregions, setting the penalty factor too large may lead to premature convergence of the population and narrow down the located search space. On the other hand, setting the penalty factor too small may render the constraint conditions ineffective, making it challenging for the population to converge. In light of this, existing research has considered the use of variable penalty factors. In the method proposed by Jiao et al. [9], the penalty factor is dynamically adjusted based on the ratio of feasible solutions in a real-time population. In the dynamic penalty function method proposed by Maldonado et al. [10], the penalty factor undergoes gradual adjustments with changes in the population generations. Furthermore, in the learning-guided parameter tuning method proposed by Fan et al. [11] an adaptively generated penalty factor significantly enhances the adaptability of the penalty factor. From this, it is evident that the penalty function method is straightforward, yet the proper setting of the penalty factor is still challenging and usually relies on human expertise.

The hybrid method bridges evolutionary algorithms with traditional optimization methods, effectively integrating the strengths of evolutionary algorithms and mathematical programming. This approach aims to maintain both population convergence and diversity, while emphasizing the feasibility of optimized solutions. In general, evolutionary algorithms guide the population towards promising regions, and mathematical programming further explores feasible solutions within the located region. Morovati et al. [12] combined evolutionary algorithms with the Zoutendijk feasible direction method to search for solutions that meet both optimization objectives and constraints. Schutze et al. [13] utilize traditional optimization methods to explore information about nearby solutions of population. They predict the subsequent evolutionary direction of the population, guiding it towards the feasible regions. The hybrid approach is a comprehensive method that combines global and local search capabilities, effectively balancing optimization objectives and constraint satisfaction. However, the timing of integrating evolutionary algorithms and mathematical programming still requires further research.

The idea behind multi-objective optimization is to transform constrained multi-objective optimization problems into unconstrained multi-objective optimization problems. In contrast to the penalty function method, this approach transforms constraints into one or more optimization objectives. Through continuous population evolution, the method optimizes the transformed objectives, aiming to reduce the degree of constraint violation. Various methods may transform constraint conditions into a different number of optimization objectives. Long et al. [14] converted all constraints into a single optimization objective related to the degree of constraint violation. Vieira et al. [15]. transformed constraint conditions into two optimization objectives: the total degree of constraint violation and the number of violated constraints. Ming et al. [16] transformed constraint conditions into three optimization objectives. Compared with the penalty function method, this method

offers greater flexibility in handling constraint conditions. However, as the number of transformed objective functions increases, the handling complexity also rises.

2.2. The Objective and Constraint Separation Method

The idea behind the objective–constraint separation method is to handle the problem's objective and constraints separately. While reducing the degree of constraint violation, it also optimizes the objective functions. The conventional methods include the constrained dominance principle (CDP) [17], the ε constrained method [18], and stochastic ranking (SR) [19].

2.2.1. Constraint Dominance Principle

In the CDP, for solutions x and y, x is considered to be non-dominated to y when any of the following conditions is satisfied:

(1) Both x and y are feasible solutions, but x has a better optimization objective value.
(2) x is a feasible solution, whereas y is an infeasible solution.
(3) Both x and y are infeasible solutions, but $cv(x) < cv(y)$, where $cv(\cdot)$ represents the degree of constraint violation for the optimization solution.

The selection principle of CDP is relatively straightforward, as it tends to retain feasible solutions and eliminate infeasible ones, leading the population to be prone to local optima.

Deb et al. [17] proposed the abovementioned constraint handling method, which to some extent alleviates the solving pressure associated with constrained multi-objective problems. Subsequently, to utilize effective information from infeasible solutions, Saha and Ray [20] introduced a probability point-based method for repairing equality constraints.

2.2.2. ε Constrained Method

In the ε constraint method, the approach involves relaxing constraints and gradually reducing the value of ε to tighten the constraints. This method utilizes information from excellent infeasible solutions to guide the population towards the feasible region during evolution. When $\varepsilon = 0$, the ε constraint method is equivalent to CDP. For solutions x and y, x is considered to be non-dominated to y when either of the following conditions is satisfied:

(1) Both x and y are ε feasible solutions, but x has a better optimization objective value.
(2) x is an ε feasible solution, whereas y is not an ε feasible solution.
(3) Both x and y are not ε feasible solutions, but $cv(x) < cv(y)$, where $cv(\cdot)$ represents the degree of constraint violation for the optimization solution.

The ε constraint method to some extent utilizes information from excellent infeasible solutions, but the choice of ε often relies on human expertise, which may still lead the population into local optima. Sana Ben Hamida et al. [21] tried to propose an ε-based algorithm by temporarily tolerating high violation degrees to expand the feasible region, and gradually narrowing the feasible region to enhance global search capability. This method effectively utilizes information from infeasible solutions, thereby guiding the population moves from infeasible regions to feasible regions, improving the algorithm's performance. In addition, the small habitat technique with an adaptive radius can be used to extend the penalty function. Ponsich et al. [22] incorporated constraint violation and a scalar function as optimization objectives, transforming CMOPs into bi-objective problems. They integrated the ε constraint method into MOEA/D to prove the performance on solving the CMOPs.

2.2.3. Stochastic Ranking

SR refers to the probabilistic selection of constraint dominance and objective dominance to better generate offspring populations. Compared with the two methods mentioned above, this approach better utilizes valuable information from excellent infeasible solutions, increases the diversity of the population, avoids premature convergence of the population,

and enhances global search capability. However, the setting of selection probabilities still relies on human expertise, which may lead to the population falling into local optima. SR has diverse applications, each of which possesses flexible performance and functionality. Liu et al. [23] combined indicator-based MOEA with CDP, the ε constraint method, and the SR method, respectively. They proposed multiple indicator-based constrained multi-objective methods, separately studying their solution performance. Jan and Khanum [24] first embedded SR in the framework of the decomposition-based multi-objective evolutionary algorithm (MOEA/D), effectively enhancing the algorithm's performance. Ying et al. [25] drew inspiration from simulated annealing and proposed an annealing SR mechanism. This mechanism can fully utilize well-behaved feasible solutions, guiding the evolution toward global optima. Sabat et al. [26] combined particle swarm optimization with SR, proposing a new hybrid algorithm for solving standard-constrained engineering design problems. The proposed hybrid algorithm leverages the domain independence of SR and the faster convergence of particle swarm optimization. To address the drawback of setting SR parameters relying on manual expertise, some scholars have proposed methods for adaptively adjusting SR parameters. Li et al. [27] adaptively randomize sorting based on the dynamic balance of convergence and diversity in the population's state in high-dimensional space. Ying et al. [28] proposed an adaptive SR mechanism by dynamically controlling probability parameters based on the difference between the current evolutionary stage and the degree of individual constraint violation. Gu et al. [29] proposed an enhanced SR strategy correlating fitness and probability operators, which comprehensively considers the convergence and diversity of the population to improve the quality of candidate solutions. There are various methods for adjusting evaluation indicators using SR, each with its characteristics. Wang et al. [30] introduced a logistic regression model to correct surrogate-assisted fitness evaluations and employed SR to further reduce the impact of the approximate constraint function. Li et al. [31] proposed a multi-objective algorithm for multi-objective optimization problems, which utilized SR techniques to balance search biases across different objectives. Chen et al. [32] proposed a multi-objective algorithm based on gradient SR. This employed a two-layer gradient SR method for offspring selection, guiding the direction of Pareto front selection and enhancing the relationships between different indicators in indicator-based MOEA.

2.3. The Multi-Operator Method

The dynamic selection of multiple reproduction operators can influence the distribution of the population in the search space. The fundamental idea of the multi-operator method is that of performing different reproduction operators on population to meet the solving requirements. The selected multiple reproduction operators need to balance well the feasibility, convergence, and diversity of the solution set. In the multi-operator algorithm proposed by Yu et al. [33], infeasible solutions are reproduced with special mutation operators from the mutation operator pool with a certain probability. In the approach proposed by Xu et al. [34], distinct mutation strategies are applied to feasible and infeasible solutions, facilitating the population's rapid evolution toward the feasible region. Liu et al. [35] divided the entire population into multiple subpopulations, with each subpopulation adopting different crossover operators. This approach facilitates the population in escaping local optima and fully exploring the search space. He et al. [36] implemented different crossover and mutation operators for feasible solutions and good infeasible solutions, driving the infeasible solutions toward the feasible region. Tian et al. [37] employed deep reinforcement learning to select different operators at different stages, striking a balance between the diversity and convergence of the population. Zuo et al. [38] also used deep reinforcement learning to dynamically employ reproduction operators, which can be embedded into existing evolutionary algorithms and improve their performance.

It can be observed that the multi-operator method can adapt to the solving requirements, balancing the feasibility, convergence, and diversity of optimized solutions. However, because the multi-operator method usually relies on manual expertise, configuring

performance-complementary reproduction operators, and determining the appropriate timing or scope of using reproduction operator poses a challenge.

2.4. Discussion

From this section, it can be inferred that the objective and constraint separation method is simple to conduct; however, using a single reproduction operator may lead to a biased search direction for the population. Therefore, it is necessary to employ multiple reproduction operators to adjust the population's search directions. The multi-operator method can enhance the population diversity but this often relies on human expertise to determine the timing and scope of using different operators. Similar to the difficulty in using the constraint separation method, reliance on human expertise is one of the limitations of the multi-operator method, and this results in a subjective determination of the timing and scope for the use of different operators. Using reinforcement learning methods can effectively reduce the dependence on human expertise. This indicates that combining reinforcement learning with the objective and constraint separation method, and the multi-operator method can leverage strengths and mitigate weaknesses, which has promising prospects for improving algorithm performance.

In light of this, a self-evolving optimization approach guided by the feasibility state of the population is proposed in Section 3, which leverages the feasibility state of population solutions within a reinforcement learning framework to recommend the reproduction operator choices for population evolution. This significantly enhances the autonomy of solving problems, avoiding the subjectivity caused by human expertise.

3. The Proposed Method

3.1. Overall Framework

This section introduces the population feasibility state guided autonomous evolutionary optimization method. It first characterizes the feasibility state of the population solutions based on both feasibility and ε feasibility of the solutions, where the individual i of population P_t is presented as $P_{i,t} = (x_1, x_2, \ldots, x_D)$. Subsequently, a real-time mapping model between the state of solutions and reproduction operator choices based on the Q-learning reinforcement learning model is constructed. Finally, based on the current state of the population, the mapping model is utilized to recommend the reproduction operator to generate offspring population. This method exhibits a significant performance improvement for constrained multi-objective optimization algorithms employing the ε constraint mechanism. By employing the aforementioned method, it becomes possible to autonomously select appropriate reproduction operators for different population states, efficiently addressing constrained multi-objective optimization problems. Q-learning is used to build the mapping relationship between population state and operators. The reason for using Q-learning is that we regard dynamically selecting operators for varying populations as a Markov decision process, and Q-learning is an excellent reinforcement learning algorithm used to solve Markov decision process problems. The ability of Q-learning includes the following aspects:

(1) Learning the optimal strategy: Q-learning can learn the optimal strategy, that is, taking actions that can maximize cumulative returns in each state. By continuously updating the Q-value table, the agent can gradually converge to the optimal strategy.
(2) No model required: Q-learning does not require modeling of the environment; that is, it does not require prior knowledge of state transition probabilities and reward functions. It learns the Q-value function through interaction with the environment, thus achieving model free learning.
(3) Adaptability: Q-learning can adapt to different environments and tasks. It can handle continuous state space and action space.
(4) Convergence: Under certain conditions, the Q-learning algorithm can converge to the optimal Q-value function. This means that the agent can learn the optimal strategy within a limited time.

An analogy for the concept of evolutionary computation is that of reinforcement learning. Population is analogous to agent, population state is analogous to agent state, operator is analogous to action, indicator-based population evaluation is analogous to reward, and fitness landscape is analogous to environment.

The overall framework of the proposed method is illustrated in Algorithm 1. Regarding existing constrained multi-objective optimization algorithms, the method proposed in this paper only requires the incorporation of two steps in the algorithm process. One is the selection of the reproduction operator; the other is Q-value learning. Specifically, begin by initializing the relevant parameters for reinforcement learning, along with the Q-table for storing Q-values and the Count-table for tracking the frequency of action selections. Then, in each generation, select the reproduction operator according to Algorithm 2 to generate offspring populations. Finally, update the Q-value table with (s_t, a_t, r_t) according to Algorithm 3. The whole flowchart of the proposed method is presented in Figure 1, where the extra modules required by this method are highlighted.

The methods for selecting the reproduction operator and updating the Q-value table are outlined in Sections B and C, respectively.

Algorithm 1 Overall framework

Input: Maximum generation T_{max}, Population size NP, Problem dimension D, Scaling factor in DE operator F, Crossover CR; reward update parameter α, reward predictability parameter γ, greedy strategy parameter ϵ.

Output: final population $P_{T_{max}}$

$t \leftarrow 1$;

Initialize population P_t with equation (4)

$P_{t-1} \leftarrow P_t$;

$s_t \leftarrow (Rf_t, Re_t)$;

$\alpha \leftarrow 0.005$, $\gamma \leftarrow 0.2$, $\epsilon \leftarrow 0.8$;

Q-table $\leftarrow 0$ and Count-table $\leftarrow 0$;

$A_i = \varepsilon_i^{(t)} + \delta$, $B_i = \dfrac{T}{cp\sqrt{\ln\left(\frac{\varepsilon_i^{(0)}+\delta}{\delta}\right)}}$;

$\varepsilon_i^{(t)} = A_i e^{-\left(\frac{t}{B_i}\right)^{cp}} - \delta$;

While $t \leq T_{max}$ do

  Generate the Mating Pool P_t;

  Select operator with Algorithm 2;

  $Q_t \leftarrow reproduction(P_t)$;

  $R_t \leftarrow Q_t \cup P_t$;

    For e = 1:$|R_t|$

      $g_i(x^e) = max\left(g_i(x^e) - \varepsilon_i^{(t)}, 0\right)$;

      $h_j(x^e) = max\left(\left|h_j(x^e)\right| - \varepsilon_j^{(t)}, 0\right)$;

      $cv(x^e) = \frac{1}{n}\left(\sum_{i=1}^{l} \frac{g_i(x^e)}{max_{x^e \in P_t}\{g_i(x^e)\}} + \sum_{j=l+1}^{n} \frac{h_i(x^e)}{max_{x^e \in P_t}\{h_i(x^e)\}}\right)$;

      $F(x^e) = (f_1(x^e), f_2(x^e), \ldots, f_m(x^e))$;

    end

  $Rank_t \leftarrow ND\_sorting(cv(R_t), F(R_t))$;

  $P_{t+1} \leftarrow R_{Rank_t\{1:NP\},t}$;

  Get$(s_t,\ a_t,\ r_t)$ with Algorithm 3 to update Q-table;

  $t \leftarrow t + 1$;

  $A_i = \varepsilon_i^{(t)} + \delta$, $B_i = \dfrac{T}{cp\sqrt{\ln\left(\frac{\varepsilon_i^{(0)}+\delta}{\delta}\right)}}$;

  $\varepsilon^{(t)} = A_i e^{-\left(\frac{t}{B_i}\right)^{cp}} - \delta$;

End while

Return $P_{T_{max}}$;

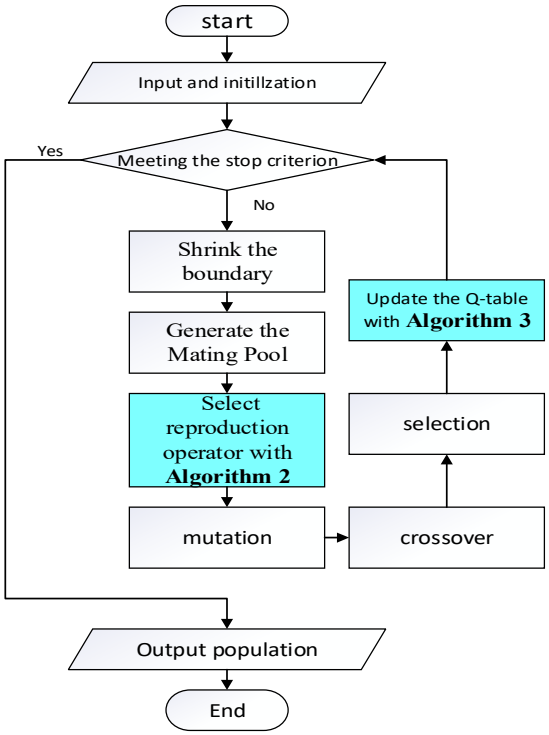


Figure 1. A flowchart of the overall framework.

3.2. The Selection of Reproduction Operator

3.2.1. Reproduction Operator for Regulating Population State

In evolutionary algorithms, regulating the population state is achieved through different reproduction operators, each with its advantages and disadvantages. Therefore, employing different reproduction operators for different population states has a significant impact on population regulation. Among various evolutionary algorithms, Differential Evolution (DE) [38] and Genetic Algorithm (GA) [9] are very typical and have been widely applied. The reproduction operator in DE typically exhibits good convergence, while the reproduction operator in GA possesses strong global search capabilities. Combining the reproduction operator of DE and GA enables a balance between global exploration and local exploitation, contributing to the improvement of the algorithm's search performance. Therefore, this paper considers using two reproduction operators from DE and GA to generate high-quality offspring populations.

Algorithm 2 Two-stage reproduction operator selection

Input: current generation t, Q-table, Count-table;
Output : operator a_t;
1. If $t \leq T_{sam}$ then
2. $a_t^{max} \leftarrow rand_{operator}$;
3. Else
4. r'_t= Q-table(s_t, a_t)/Count-table(s_t, a_t);
5. $a_t^{max} \leftarrow arg\max\{r'_t\}$;
6. End if
7. If rand $<$ ϵ then
8. $a_t^{max} \leftarrow rand_{operator}$;
9. End if

With this approach, there are two reproduction operators used for regulating the population state. Let us denote a_t as the identifier of the reproduction operator adopted by population P_t. The possible values and their meanings are as follows: $a_t = 1$ represents the

execution of the reproduction operator from DE, and $a_t = 2$ represents the execution of the reproduction operator from GA.

3.2.2. Two-Stage Reproduction Operator Selection

In the early stage of the evolutionary search, due to the unavailability of sufficient training samples, the Q-value table (Q-table) cannot provide effective guidance on reproduction operator choices. Therefore, the evolution of the population is divided into two stages. First, from the 1^{st} generation to the T_{sam} generation, it is the reinforcement learning warm-up stage, where samples are collected for updating the Q-table. Second, from the $T_{sam} + 1$ generation until the end of the population evolution, it is the application stage of reinforcement learning. This two-state operator selection assisted by Q-learning can be visualized in Figure 2, and its implementation details are illustrated in Algorithm 2. In Algorithm 2, the first stage involves randomly selecting the reproduction operator a_t to accumulate training samples and update the Q-table. In the application stage, for the evolutionary population P_t at generation t, the state s_t of this population is used as input to the Q-table to obtain performance predictions for different reproduction operators. Denote the performance prediction obtained after applying the reproduction operator a_t to P_t as r'_t. For all reproduction operators, let

$$a_t^{max} = arg\ \max\{r'_t\} \tag{1}$$

denote the reproduction operator with the maximum performance prediction. So, a_t^{max} is the reproduction operator adopted for the subsequent evolution of the population.

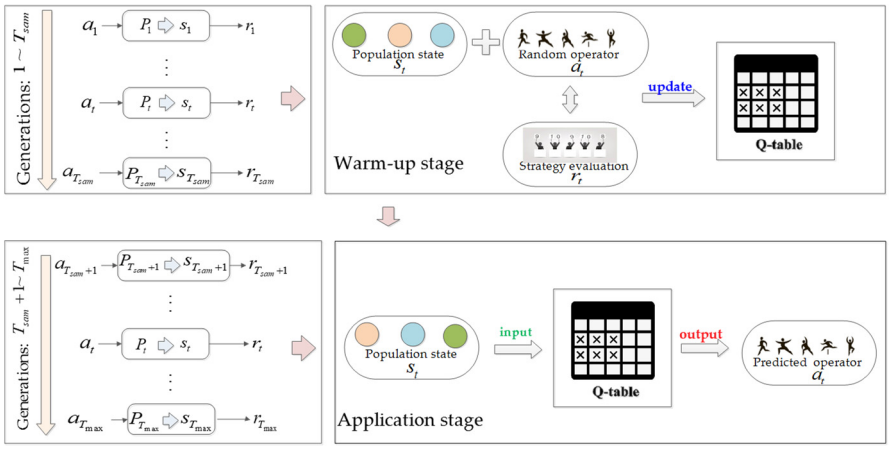


Figure 2. The embedded evolutionary algorithm and Q-learning.

It is worth noting that the frequency of choosing each reproduction operator has a significant impact on the Q-table updates and may result in inaccurate predicted Q-values. Therefore, this paper also records the frequency of reproduction operator choices in the Count-table to calculate the average Q-value for each population state s_t, aiming to improve the accuracy of Q-value predictions and ensure the stability of algorithm performance. Hence, the abovementioned r'_t is obtained by

$$r'_t = \text{Q-table}/\text{Count-table} \tag{2}$$

To mitigate the negative impact of overfitting in Q-learning and enhance the exploration capability of the population state space, this paper borrows the Epsilon–Greedy strategy. With a small probability ϵ, a strategy is randomly chosen from the reproduction operator space instead of the one recommended by the Q-table. This operator is then

used for the subsequent evolution of the population. The ϵ based operator selection is presented by

$$a_t^{max} = \begin{cases} \text{argmax}\{r_t'\} & if\ rand > \epsilon \\ rand_{operator} & otherwise \end{cases} \tag{3}$$

where $rand_{operator}$ means a random operator is selected.

3.3. Update the Q-Table

The updating of the Q-table is illustrated in Algorithm 3. Firstly, calculate the proportion of feasible and ε feasible solutions in the population to obtain s_t. Then, calculate r_t by comparing P_{t-1} and P_t. Finally, update the Q-value table based on the adopted a_t and record the update frequency in the corresponding position of the Count-table.

3.3.1. Characterizing the Population State Based on the Feasibility of Population

In constrained multi-objective evolutionary algorithms (CMOEAs), a x representing $D-dimensional$ decision variables is regarded as a population individual, and the e-th population individual is denoted as x^e. Hence, a population with NP individuals in the t-th generation is denoted as $P_t = \left(x^1, x^2, \ldots, x^e, \ldots, x^{NP}\right)$, where $x^e = \left(x_1^e, x_2^e, \ldots, x_d^e, \ldots, x_D^e\right) \in R^D$. x_d^e is initialized by

$$x_d^e = rand(LB_d, UB_d), d = 1, 2, ..., D \tag{4}$$

LB_d and UB_d are lower and upper boundaries, respectively. During the evolution of P_t, its feasibility and ε-feasibility are two important pieces of knowledge during the population evolution process. Both of them influence the search direction and efficiency of the algorithm. In view of this, the proportions of both can be used to jointly characterize the population state. As introduced in DCNSGA-II, the normalized average degree of the constraint violation of P_t on all constraints is regarded as the constraint violation objective by

$$cv(x^e) = \frac{1}{n}\left(\sum_{i=1}^{l} \frac{g_i(x^e)}{max_{x^e \in P_t}\{g_i(x^e)\}} + \sum_{j=l+1}^{n} \frac{h_i(x^e)}{max_{x^e \in P_t}\{h_i(x^e)\}}\right) \tag{5}$$

where $g_i(x^e) = max\left(g_i(x^e) - \varepsilon_i^{(t)}, 0\right)$, $h_j(x^e) = max\left(\left|h_j(x^e)\right| - \varepsilon_j^{(t)}, 0\right)$, $\varepsilon_i^{(t)}$ and $\varepsilon_j^{(t)}$ are the dynamic constraint boundaries, $t = 0, 1, ..., T$, T is the maximum number of generations. $\varepsilon_i^{(t)}$ and $\varepsilon_j^{(t)}$ are updated following the same criterion. As an example, $\varepsilon_i^{(t)}$ is updated by

$$\varepsilon_i^{(t)} = A_i e^{-\left(\frac{t}{B_i}\right)^{cp}} - \delta,\ i = 1, 2, \cdots, q \tag{6}$$

where δ is a close-to-zero value ($\delta = 1 \times 10^{-8}$) and cp controls the decreasing trend of ε. A_i and B_i are updated as follows:

$$A_i = \varepsilon_i^{(t)} + \delta,\ B_i = \frac{T}{cp\sqrt{\ln\left(\frac{\varepsilon_i^{(0)} + \delta}{\delta}\right)}} \tag{7}$$

With $cv(x^e)$, the constraint violation degree x^e is computed. Thus, the population state can be characterized. Firstly, calculate the proportions of feasibility

$$Rf_t = \frac{FNP_t}{NP} \tag{8}$$

and ε-feasibility in the population by

$$Re_t = \frac{ENP_t}{NP} \tag{9}$$

where NP is the population size, FNP and ENP are number of feasible population and ε-feasible population. Next, divide the proportions of both in the population into three parts, respectively, including zero to one-third, one-third to two-thirds, and two-thirds to one. Consider that feasibility proportion is not higher than the ε-feasibility proportion, so the population P_t only contains six states, denoted as

$$s_t = (Rf_t, Re_t) \tag{10}$$

Algorithm 3 Update the Q-table

Input: s_t, a_t, P_{t-1}, P_t, parameters of Q-learning, Count-table, Q-table;
Output: Q-table, Count-table;
1. $s_t \leftarrow (Rf_t, Re_t)$;
2. If $\phi_{t-1} > 0 || \phi_t > 0$ then
3. $r_t = \frac{\phi_{t-1} - \phi_t}{\phi_{t-1}}$;
4. Else
5. $r_t = \frac{HV_t - HV_{t-1}}{HV_{t-1}}$;
6. End if
7. $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot max(Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t))$;
8. Count-table $(s_t, a_t) \leftarrow$ Count-table$(s_t, a_t) + 1$;

3.3.2. Stage-Wise Evaluation of Population Solutions

This paper adopts a stage-wise evaluation method. If the evolved population has no feasible solutions, the performance of reproduction operators is evaluated based on the extent to which the constraint violation of the best individual in the population increases or decreases. Considering the best individual in population P_t, apply the strategy of this population as a_t. To evaluate the performance of a_t, calculate the violation degree of this individual for each constraint, summing up these violation degrees to obtain the overall constraint violation degree ϕ_t. Then, the performance of the reproduction operator a_t, denoted as r_t, can be expressed as

$$r_t = \frac{\phi_{t-1} - \phi_t}{\phi_{t-1}} \tag{11}$$

It can be observed that the values of r_t fall within the range of 0 to 1. It is worth noting that $r_t = 0$ means applying the reproduction operator a_t does not lead to an improvement in the population's satisfaction with constraints. $r_t = 1$ means applying a_t, the best individual in the population becomes a feasible solution.

When the evolved population contains one or more feasible solutions to the problem, the performance of reproduction operators is evaluated based on the extent to which the Hypervolume (HV) indicator of feasible solutions improves. The Hypervolume (HV) not only reflects the diversity of feasible solutions but also indicates the convergence of these solutions. For the population P_t, calculate the HV indicator of feasible solutions in this population, denoted as HV_t. Then, the performance r_t of reproduction operator a_t can be expressed as

$$r_t = \frac{HV_t - HV_{t-1}}{HV_{t-1}} \tag{12}$$

HV_t measures the volume of the objective space enclosed by the obtained solution set in P_t and a reference point z^r, which can be obtained by

$$HV_t = VOL(\bigcup_{P_{m,t} \in P_t} [P_{1,t}, z_1^r] \times \cdots \times [P_{m,t}, z_m^r]) \tag{13}$$

where VOL represents the Lebesgue measure. With this approach, the performance r_t of the reproduction operator a_t can be expressed as

$$r_t = \begin{cases} \frac{\phi_{t-1}-\phi_t}{\phi_{t-1}}, & if \ \phi_t > 0 \\ \frac{HV_t-HV_{t-1}}{HV_{t-1}}, & otherwise \end{cases} \tag{14}$$

3.3.3. Q-Value Update

This paper employs the Q-value function for updating the Q-value table, and its functional expression is as follows:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot max(Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t)) \tag{15}$$

where α is the learning rate controlling the willingness to accept newly observed information. r_t is the immediate reward obtained after taking action a_t. γ is the discount factor, indicating the importance given to future rewards. s_{t+1} is the next state transitioned to after taking action a_t.$max(Q(s_{t+1}, a_{t+1}))$ represents the Q-value corresponding to choosing the optimal action in the next state s_{t+1}. By constantly interacting with the population evaluation function, the population can choose an operator based on the current state and update the corresponding Q-value in the Q-table based on the immediate return and the maximum expected return value of the next state. Through multiple iterations, the values in the Q-table will gradually converge to the optimal value, and the agent can choose the optimal action based on the values in the Q-table to achieve the optimal strategy.

3.4. Further Explanation

The research [37] presented by Tian et al. is enlightening on guiding the operator selection with deep reinforcement learning. Although we also use reinforcement learning to solve the same problem, our proposed method is still different from their method in the following aspects:

(1) The problem type aimed at is different. Our research is devoted to solving constrained multi-objective problems, while the mentioned research is proposed for the unconstrained optimization problems.
(2) The definition of population state is different. Our research defines the population state with the population feasibility information, which is useful to reduce the complexity of the mapping model between state, action, and reward. This option is important for proposing an efficient online adaptive operator selection algorithm, without the offline training phase. Oppositely, the population state in the abovementioned paper is population individual. Hence, their mapping model will be more accurate once well trained. However, it is not convenient to transfer the mapping model between the problems with different dimensions.
(3) The definition of reward function is different. Our research defines the reward function using the indicator-based evaluation method, while the abovementioned paper employs the population fitness method to evaluate the performed action. Oppositely, their method has low cost on the reward function, while our method is more stable on evaluating the effectiveness of the performed operator.

In addition, our proposed approach reduces the degree of reliance on human experience. Although the new approach still relies on human experience, it still makes sense to do so. This is because the elimination of difficult-to-control parameters and the introduction of new parameters can help simplify algorithm design, and improve algorithm stability and controllability. By reducing the parameters that are difficult to control, the complexity of the algorithm can be significantly reduced, the uncertainty will be alleviated, and the reliability of the algorithm is finally improved. At the same time, introducing new parameters and ensuring that they are less sensitive can better control the behavior of the algorithm, making it easier to adjust and optimize the algorithm. This can improve the performance and efficiency of the algorithm, and make the algorithm easier to apply and popularize.

3.5. Complexity Analysis

A review of the content of the above introduction shows that the time complexity of the proposed method is mainly influenced by the extraction of population feasibility states, the execution of evolutionary strategies, and the evaluation of evolutionary strategy performance, as well as the warm-up and application stages of Q-learning.

When solving an optimization problem with a decision space of m dimensions, for an evolutionary algorithm with a maximum number of generations set to T_{max} and a population size of n, extracting the population states involves the necessity to compute the feasibility and ε feasibility of the population for T_{max} generations. Executing evolutionary strategies involves vector calculations for all individuals in the population. The evaluation of evolutionary strategies is related to the number of feasible solutions and the number of optimization objectives, denoted as m. For Q-learning, it is necessary to compute the number of Q-table updates for each generation. The time complexities for these four components are as follows:

(1) The time complexity of extracting population states is $O(T_{max}nd)$.
(2) The time complexity of executing evolutionary strategies in the worst-case scenario is $O(T_{max}nd)$.
(3) The time complexity of evaluating evolutionary strategies in the worst-case scenario is $O\left(T_{max}n^2m + T_{max}n^{m-2}logn\right)$.
(4) The time complexity of pre-warming and applying Q-learning in the worst-case scenario is $O(2T_{max})$.

From this, it can be inferred that the time complexity of the proposed method is primarily determined by T_{max}, n, d, m, and the number of Q-table updates. For large-scale constrained multi-objective optimization problems, the above time complexity is primarily determined by the extraction of population states and the execution of evolutionary strategies, which is $O(T_{max}nd)$. When the number of objectives increases, the time complexity is primarily determined by the evaluation of evolutionary strategies, which is $O\left(T_{max}n^2m + T_{max}n^{m-2}logn\right)$.

4. Experiment

To evaluate the performance of the proposed method in this paper, four groups of experiments were conducted in this section. The first group of experiments compared the performance of the DCNSGA-III [39] with that embedded our method, named IDCNSGA-III. This aimed to validate the effectiveness of the proposed method on benchmark test problems. The second group of experiments compared the IDCNSGA-III with five superior-performing algorithms for constrained multi-objective optimization. This further elucidated the performance of the proposed method. The third experiment compared our algorithm with the ε-based multi-objective optimization algorithms, which illustrates the advances achieved in our algorithms. The fourth experiment studied the influence of parameter setting on algorithm performance, which can help to recommend the best parameter setting. The experimental setup included the following environmental conditions: 13th Gen Intel(R) Core(TM) i5-13400 2.50 GHz, 16 GB RAM, Windows 10, and PlatEMO [40].

4.1. Benchmark Test Suits

In terms of test problems, the experiments utilized four groups of benchmark test problems: LIR-CMOP, MW, TREE, and RWMOP [41]. The four groups of test problems comprehensively cover four common types of constrained multi-objective optimization problems [42], namely: (1) the UPF and the CPF completely coincides; (2) parts of the UPF are feasible, and the CPF is part of the UPF; (3) some regions of the UPF are feasible, and the CPF and UPF partly coincide; (4) the UPF is located in the infeasible region, and the CPF is wholly separated from the UPF. It can be seen that the selected test problems comprehensively assess the method proposed in this paper.

4.2. Comparison Algorithms and Parameters Setting

To test the effectiveness of the proposed method, we chose the algorithm DCNSGA-III as the original algorithm embedding the proposed method. The algorithm embedded with the proposed method is named IDCNSGA-III, which is compared with DCNSGA-III to experimentally evaluate its effectiveness. The parameter of DCNSGA-III is kept consistent with the original literature. For IDCNSGA-III, the greedy strategy factor is set to $\epsilon = 0.8$, and the parameters for Q-learning are set as $\gamma = 0.2$, $\alpha = 0.005$. $T_{sam} = 0.2 \times T_{max}$, $T_{upd} = 0.1 \times T_{max}$.

To further illustrate the performance of the proposed method, a comparison was conducted with CMOEA-MS [43], BiCo [44], AGE-MOEA-II [45], DSPCMDE [46], NSGA-II [17], IDCNSGA-III, Trip [47], DPPPS [47], and CAEAD [48]. The parameter values for the aforementioned comparative algorithms were kept consistent with the PlatEMO.

The population size for all test problems was set to 50, and the number of function evaluations for each test problem was 10,000.

4.3. Performance Indicator

The HV is a comprehensive and effective indicator to evaluate the convergence and diversity of algorithms. In this paper, it was chosen as the evaluation criterion for algorithm performance. The symbols "+", "−", and "=" are used to indicate that an algorithm significantly outperforms, underperforms, or has no significant difference compared to another algorithm, respectively.

4.4. Effectiveness of the Proposed Method

For each test problem, DCNSGA-III and IDCNSGA-III were run independently 30 times, obtaining HV indicator values. The HV results statistics are presented in Table 1, where the best results are highlighted. For the HV indicator, in 17 out of 33 test problems, the performance of IDCNSGA-III was significantly better than the original algorithm. From this, it can be concluded that applying the method proposed in this paper significantly improves the algorithm's performance.

The TREE testing problem has a high dimensionality, and it poses a considerable challenge to be solved. In the comparison, IDCNSGA-III demonstrated a clear advantage. This means that, although the test problems have high dimensionalities, the fitness landscapes are relatively simple. IDCNSGA-III can easily acquire effective evolutionary information, guiding the direction of population movement and promoting rapid convergence of the population.

The feasible region of the LIR-CMOP test problem is very small. Some problems exhibit only one curve, and the shape of the constraint Pareto front consists of several disjointed segments or sparse points. IDCNSGA-III exhibits significant advantages on the majority of LIR-CMOP problems. There is no apparent advantage for IDCNSGA-III on LIR-CMOP7 to LIR-CMOP8 problems, which may be caused by the relatively low solving difficulty of these problems. In this case, the algorithm can easily obtain feasible solutions, leading to a limited regulatory effect for the proposed method. There is also no significant advantage for IDCNSGA-III on LIR-CMOP13 to LIR-CMOP14, which are two three-dimensional problems. This may be caused by the high difficulty in solving these problems, and the algorithm struggles to obtain feasible solution information over the long term, resulting in less noticeable performance in regulating the population.

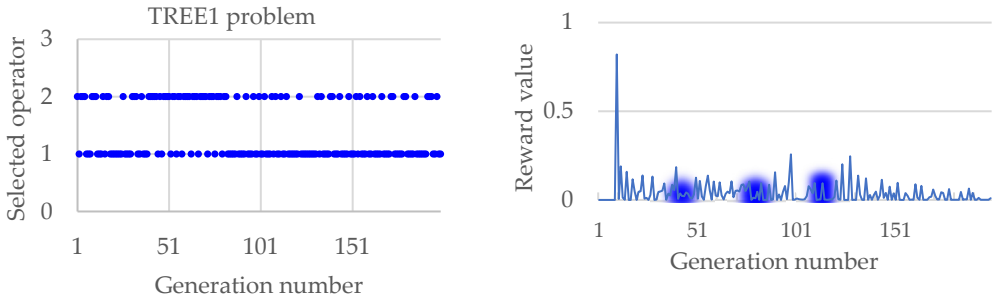
Table 1. HV indicator statistics for IDNSGA-III and DCNSGA-III.

Problem	M	D	IDCNSGA-III	DCNSGA-III
TREE1	2	300	7.2952×10^{-1} (1.64×10^{-2}) +	7.0269×10^{-1} (7.73×10^{-3})
TREE2	2	300	7.6076×10^{-1} (1.01×10^{-2}) +	7.4026×10^{-1} (6.86×10^{-3})
TREE3	2	120	7.9388×10^{-1} (9.08×10^{-3}) +	7.6790×10^{-1} (1.20×10^{-2})
TREE4	2	120	7.0300×10^{-1} (3.38×10^{-2}) +	6.3061×10^{-1} (3.22×10^{-2})
TREE5	2	120	7.8099×10^{-1} (2.67×10^{-2}) +	7.4329×10^{-1} (2.03×10^{-2})
LIRCMOP1	2	30	1.2444×10^{-1} (2.56×10^{-2}) +	1.0638×10^{-1} (2.28×10^{-2})
LIRCMOP2	2	30	2.4912×10^{-1} (2.00×10^{-2}) +	2.2259×10^{-1} (2.44×10^{-2})
LIRCMOP3	2	30	1.0955×10^{-1} (2.05×10^{-2}) +	9.0574×10^{-2} (1.32×10^{-2})
LIRCMOP4	2	30	2.0854×10^{-1} (1.79×10^{-2}) +	1.8894×10^{-1} (1.89×10^{-2})
LIRCMOP5	2	10	2.8561×10^{-1} (1.65×10^{-3}) +	2.1053×10^{-1} (2.88×10^{-2})
LIRCMOP6	2	10	1.9183×10^{-1} (1.32×10^{-3}) +	1.1070×10^{-1} (4.64×10^{-2})
LIRCMOP7	2	30	2.1168×10^{-1} (5.83×10^{-2}) =	1.9214×10^{-1} (7.85×10^{-2})
LIRCMOP8	2	30	1.3642×10^{-1} (9.86×10^{-2}) =	1.4752×10^{-1} (9.08×10^{-2})
LIRCMOP9	2	30	2.5944×10^{-1} (7.32×10^{-2}) +	1.0893×10^{-1} (3.78×10^{-2})
LIRCMOP10	2	30	1.9340×10^{-1} (1.40×10^{-1}) +	5.9978×10^{-2} (3.49×10^{-2})
LIRCMOP11	2	30	2.7786×10^{-1} (8.43×10^{-2}) +	1.6002×10^{-1} (4.40×10^{-2})
LIRCMOP12	2	30	4.0423×10^{-1} (7.72×10^{-2}) +	2.8276×10^{-1} (1.00×10^{-1})
LIRCMOP13	3	30	$0.0000 \times 10^{+0}$ $(0.00 \times 10{+}0)$ =	3.7687×10^{-6} (2.06×10^{-5})
LIRCMOP14	3	30	1.7358×10^{-5} (5.94×10^{-5}) =	1.6225×10^{-5} (6.98×10^{-5})
MW1	2	30	1.4722×10^{-1} (1.30×10^{-1})	Infeasible
MW2	2	30	3.8124×10^{-1} (1.29×10^{-1}) =	4.3084×10^{-1} (8.17×10^{-2})
MW3	2	30	5.0566×10^{-1} (1.43×10^{-2}) =	4.3151×10^{-1} (1.60×10^{-1})
MW4	3	30	4.3348×10^{-1} (5.15×10^{-2})	Infeasible
MW5	2	30	1.1962×10^{-1} (7.62×10^{-2})	Infeasible
MW6	2	10	2.9226×10^{-1} (1.89×10^{-2}) =	2.9375×10^{-1} (3.12×10^{-2})
MW7	2	10	4.0304×10^{-1} (1.59×10^{-3}) =	4.0119×10^{-1} (3.99×10^{-3})
MW8	3	10	4.8872×10^{-1} (2.38×10^{-2}) -	5.0081×10^{-1} (1.98×10^{-2})
MW9	2	30	1.3763×10^{-1} (1.70×10^{-1}) =	1.4546×10^{-1} $(0.00 \times 10^{+0})$
MW10	2	30	2.2475×10^{-1} (9.60×10^{-2}) =	2.6132×10^{-1} (1.02×10^{-1})
MW11	2	30	4.3830×10^{-1} (1.25×10^{-3}) +	4.3107×10^{-1} (4.68×10^{-3})
MW12	2	30	1.4185×10^{-1} (2.45×10^{-1}) =	$0.0000 \times 10^{+0}$ $(0.00 \times 10^{+0})$
MW13	2	30	2.9592×10^{-1} (8.84×10^{-2}) −	3.6448×10^{-1} (5.60×10^{-2})
MW14	3	30	1.1746×10^{-1} (4.12×10^{-2}) +	6.9888×10^{-2} (4.25×10^{-2})
+/−/=			17/2/11	

The constraints in the MW test problem have various forms. Its characteristic is that the objective space is divided from a very large infeasible region into very small feasible regions, and the constrained Pareto front includes multiple isolated solutions. On MW1, MW4, and MW5, the DCNSGA-III fails to find feasible solutions, while IDCNSGA-III can find and locate feasible solutions in the space. This demonstrates that applying our method can enhance the algorithm's performance. In the case of MW8 with a high-dimensional objective space, the IDCNSGA-III's performance degrades. This may be caused by the greediness of IDCNSGA-III, which favors continuously executing a single operator that returns higher rewards, leading to premature convergence of the population. In high-dimensional problems, there are many local optima, exacerbating the issues related to the algorithm's greediness. On MW13, there is also a decline in the algorithm's performance. Because our method depends on well-defined infeasible solutions to provide effective information, however, the shape of the constraint region in MW13 is excessively narrow, providing insufficient feedback information, which leads to the ineffectiveness of the method proposed in this paper.

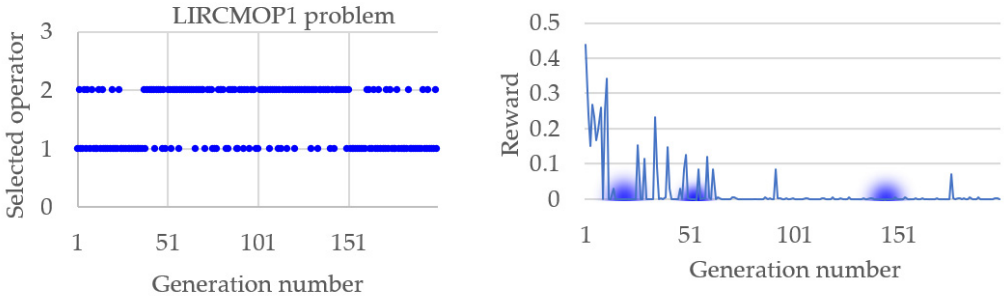
In summary, the autonomous evolution optimization method guided by the population feasibility state is effective in improving the performance of evolutionary algorithms with the ε constrained method. The autonomy here is the dynamic selection of operators realized by reinforcement learning, as shown in Figures 3 and 4, where the TREE1 and

LIRCMOP1 problems are performed as examples, and the variation of reward value along the operator selection is also simulated. In Figure 3, it can be seen that at the three key time points where the operator is preferentially selected, the reward value is obviously improved. A similar phenomenon also happened in Figure 4; at the three key time points where the operator is preferentially selected, the reward value is also obviously improved. Overall, the method proposed in this article can uniformly select operators in the early stages of evolution, accumulating sufficient and evenly distributed sampling data. In the later stage of evolution, this method can continuously adjust the operators adopted to improve the solving performance of the algorithm based on operator evaluation and real-time population state.



(1) The dynamic selection of operators (2) Reward value

Figure 3. The selected operators along the evolutionary process and the corresponding reward change in solving the TREE1 problem.



(1) The dynamic selection of operators (2) Reward value

Figure 4. The selected operators along the evolutionary process and the corresponding reward change in solving the LIRCMOP1 problem.

The demonstrated effectiveness of using reinforcement learning to autonomously select multiple operators in conjunction with the ε constrained method can be explained as follows:

(1) Collaboration of multiple operators can increase the population diversity. When solving constrained multi-objective optimization problems, the complexity of the problem often leads to the algorithm falling into local optima. By using multiple operators to work together, the population diversity is increased, thereby enlarging the coverage on search space, and helping to avoid getting stuck in local optima.
(2) Multiple operators' collaboration can improve the convergence speed of algorithms. When solving constrained multi-objective optimization problems, the algorithm needs to find the optimal solution within a limited number of iterations. By using multiple operators to work together, it is possible to search in different directions, thereby accelerating the convergence speed of the algorithm.
(3) Collaboration of multiple operators can increase the search ability of algorithms. When solving constrained multi-objective optimization problems, the algorithm needs to find the optimal solution in the search space. By using multiple operators to work together,

it is possible to search in different directions simultaneously, thereby increasing the search capability of the algorithm and helping to find better solutions.

4.5. The Comprehensive Performance of the Proposed Method

4.5.1. Compared with State-of-the-Art Constrained Optimization Algorithms

This section selects the state-of-the-art constrained multi-objective optimization algorithms—including CMOEA-MS, BiCo, AGE-MOEA-II, DSPCMDE, and the classic algorithm NSGA-II—to compare with IDCNSGA-III. Regarding the test problems, IDCNSGA-III and the selected comparative algorithms were independently run 30 times, obtaining HV indicator values. The performance statistics of HV are presented in Tables 2 and 3, where the best results are highlighted.

Table 2. HV indicator statistics for IDNSGA-III and comparison algorithms (part 1).

Problem	M	D	CMOEA-MS	BiCo	IDCNSGA-III
TREE1	2	300	6.9857×10^{-1} (8.47×10^{-3}) −	6.6672×10^{-1} (1.07×10^{-2}) −	7.2952×10^{-1} (1.64×10^{-2})
TREE2	2	300	7.3411×10^{-1} (6.04×10^{-3}) −	7.0974×10^{-1} (6.64×10^{-3}) −	7.6076×10^{-1} (1.01×10^{-2})
TREE3	2	120	7.2314×10^{-1} (3.43×10^{-2}) −	6.7538×10^{-1} (2.71×10^{-2}) −	7.9388×10^{-1} (9.08×10^{-3})
TREE4	2	120	5.8701×10^{-1} (7.96×10^{-2}) −	3.6920×10^{-1} (5.26×10^{-2}) −	7.0300×10^{-1} (3.38×10^{-2})
TREE5	2	120	7.1495×10^{-1} (5.06×10^{-2}) −	5.8449×10^{-1} (3.32×10^{-2}) −	7.8099×10^{-1} (2.67×10^{-2})
LIRCMOP1	2	30	9.7178×10^{-2} (1.24×10^{-2}) −	1.1009×10^{-1} (8.09×10^{-3}) −	1.2444×10^{-1} (2.56×10^{-2})
LIRCMOP2	2	30	2.1006×10^{-1} (1.81×10^{-2}) −	2.2736×10^{-1} (1.08×10^{-2}) −	2.4912×10^{-1} (2.00×10^{-2})
LIRCMOP3	2	30	9.0817×10^{-2} (9.06×10^{-3}) −	1.0056×10^{-1} (1.06×10^{-2}) −	1.0955×10^{-1} (2.05×10^{-2})
LIRCMOP4	2	30	1.8370×10^{-1} (1.16×10^{-2}) −	1.9512×10^{-1} (1.11×10^{-2}) −	2.0854×10^{-1} (1.79×10^{-2})
LIRCMOP5	2	10	1.1219×10^{-1} (1.01×10^{-1}) −	1.8104×10^{-2} (5.57×10^{-2}) −	2.8561×10^{-1} (1.65×10^{-3})
LIRCMOP6	2	10	8.2188×10^{-2} (5.35×10^{-2}) −	1.6302×10^{-2} (3.54×10^{-2}) −	1.9183×10^{-1} (1.32×10^{-3})
LIRCMOP7	2	30	2.3937×10^{-2} (6.37×10^{-2}) −	$0.0000 \times 10^{+0}$ $(0.00 \times 10^{+0})$ −	2.1168×10^{-1} (5.83×10^{-2})
LIRCMOP8	2	30	$0.0000 \times 10^{+0}$ $(0.00 \times 10^{+0})$−	$0.0000 \times 10^{+0}$ $(0.00 \times 10^{+0})$ −	1.3642×10^{-1} (9.86×10^{-2})
LIRCMOP9	2	30	1.0132×10^{-1} (2.56×10^{-2}) −	8.6803×10^{-2} (1.68×10^{-2}) −	2.5944×10^{-1} (7.32×10^{-2})
LIRCMOP10	2	30	5.5279×10^{-2} (2.33×10^{-2}) −	5.3817×10^{-2} (1.60×10^{-2}) −	1.9340×10^{-1} (1.40×10^{-1})
LIRCMOP11	2	30	1.6227×10^{-1} (3.12×10^{-2}) −	1.4853×10^{-1} (2.92×10^{-2}) −	2.7786×10^{-1} (8.43×10^{-2})
LIRCMOP12	2	30	1.9414×10^{-1} (6.64×10^{-2}) −	2.1506×10^{-1} (8.50×10^{-2}) −	4.0423×10^{-1} (7.72×10^{-2})
LIRCMOP13	3	30	7.8946×10^{-5} (9.93×10^{-5}) +	3.2686×10^{-5} (7.50×10^{-5}) +	$0.0000 \times 10^{+0}$ $(0.00 \times 10^{+0})$
LIRCMOP14	3	30	3.3711×10^{-4} (3.44×10^{-4}) +	1.7965×10^{-4} (2.07×10^{-4}) +	1.7358×10^{-5} (5.94×10^{-5})
MW1	2	30	Infeasible	Infeasible	1.4722×10^{-1} (1.30×10^{-1})
MW2	2	30	4.0040×10^{-1} (1.16×10^{-1}) =	4.7769×10^{-1} (8.73×10^{-2}) +	3.8124×10^{-1} (1.29×10^{-1})
MW3	2	30	3.8630×10^{-1} (1.67×10^{-1}) −	4.3716×10^{-1} (6.28×10^{-2}) −	5.0566×10^{-1} (1.43×10^{-2})
MW4	3	30	Infeasible	Infeasible	4.3348×10^{-1} (5.15×10^{-2})
MW5	2	30	Infeasible	$0.0000 \times 10^{+0}$ $(0.00 \times 10^{+0})$ =	1.1962×10^{-1} (7.62×10^{-2})
MW6	2	10	2.9209×10^{-1} (2.72×10^{-2}) =	3.0992×10^{-1} (1.04×10^{-2}) +	2.9226×10^{-1} (1.89×10^{-2})
MW7	2	10	3.9338×10^{-1} (2.98×10^{-2}) −	4.0454×10^{-1} (2.21×10^{-3}) +	4.0304×10^{-1} (1.59×10^{-3})
MW8	3	10	4.8467×10^{-1} (7.40×10^{-2}) =	5.1693×10^{-1} (1.55×10^{-2}) +	4.8872×10^{-1} (2.38×10^{-2})
MW9	2	30	Infeasible	Infeasible	1.3763×10^{-1} (1.70×10^{-1})
MW10	2	30	2.5155×10^{-1} (8.48×10^{-2}) =	2.8290×10^{-1} (8.87×10^{-2}) +	2.2475×10^{-1} (9.60×10^{-2})
MW11	2	30	3.4707×10^{-1} (6.92×10^{-2}) −	3.2792×10^{-1} (8.55×10^{-2}) −	4.3830×10^{-1} (1.25×10^{-3})
MW12	2	30	Infeasible	Infeasible	1.4185×10^{-1} (2.45×10^{-1})
MW13	2	30	5.5586×10^{-2} (1.14×10^{-1}) −	2.4713×10^{-1} (7.46×10^{-2}) −	2.9592×10^{-1} (8.84×10^{-2})
MW14	3	30	7.1004×10^{-2} (4.95×10^{-2}) −	3.5529×10^{-2} (2.09×10^{-2}) −	1.1746×10^{-1} (4.12×10^{-2})
+/−/=			2/22/4	7/21/1	

The problems in the TREE test set have a high dimensionality, making them challenging to solve. In comparison with other algorithms, IDCNSGA-III has achieved significant advantages. This may be attributed to the fact that, although the test problems have high dimensions, the fitness landscape is relatively simple, and the gradient information is clear. The algorithm can effectively utilize feedback information, promoting rapid convergence of the population.

Table 3. HV indicator statistics for IDNSGA-III and comparison algorithms (part 2).

Problem	AGEMOEA-II	DSPCMDE	NSGA-II	IDCNSGA-III
TREE1	6.6896×10^{-1} (7.51×10^{-3}) −	7.1916×10^{-1} (1.06×10^{-2}) −	6.6897×10^{-1} (8.27×10^{-3}) −	7.2952×10^{-1} (1.64×10^{-2})
TREE2	7.0870×10^{-1} (8.40×10^{-3}) −	7.5347×10^{-1} (7.02×10^{-3}) −	7.0990×10^{-1} (7.99×10^{-3}) −	7.6076×10^{-1} (1.01×10^{-2})
TREE3	7.0231×10^{-1} (2.60×10^{-2}) −	7.7027×10^{-1} (1.83×10^{-2}) −	7.0375×10^{-1} (2.25×10^{-2}) −	7.9388×10^{-1} (9.08×10^{-3})
TREE4	4.9408×10^{-1} (5.11×10^{-2}) −	6.3215×10^{-1} (3.56×10^{-2}) −	4.6792×10^{-1} (6.48×10^{-2}) −	7.0300×10^{-1} (3.38×10^{-2})
TREE5	6.5838×10^{-1} (3.60×10^{-2}) −	7.2691×10^{-1} (2.81×10^{-2}) −	6.5426×10^{-1} (3.35×10^{-2}) −	7.8099×10^{-1} (2.67×10^{-2})
LIRCMOP1	1.0273×10^{-1} (7.69×10^{-3}) −	1.2727×10^{-1} (3.16×10^{-2}) =	1.0180×10^{-1} (8.88×10^{-3}) −	1.2444×10^{-1} (2.56×10^{-2})
LIRCMOP2	2.0970×10^{-1} (1.44×10^{-2}) −	2.4769×10^{-1} (4.96×10^{-2}) =	2.1462×10^{-1} (1.32×10^{-2}) −	2.4912×10^{-1} (2.00×10^{-2})
LIRCMOP3	9.4261×10^{-2} (8.89×10^{-3}) −	1.0803×10^{-1} (2.32×10^{-2}) =	9.1757×10^{-2} (6.70×10^{-3}) −	1.0955×10^{-1} (2.05×10^{-2})
LIRCMOP4	1.8121×10^{-1} (1.18×10^{-2}) −	2.0798×10^{-1} (2.78×10^{-2}) =	1.8629×10^{-1} (1.39×10^{-2}) −	2.0854×10^{-1} (1.79×10^{-2})
LIRCMOP5	2.7174×10^{-2} (6.09×10^{-2}) −	2.8432×10^{-1} (5.36×10^{-3}) =	2.0243×10^{-2} (6.18×10^{-2}) −	2.8561×10^{-1} (1.65×10^{-3})
LIRCMOP6	2.1490×10^{-2} (4.01×10^{-2}) −	1.9216×10^{-1} (7.19×10^{-4}) =	1.5007×10^{-2} (3.17×10^{-2}) −	1.9183×10^{-1} (1.32×10^{-3})
LIRCMOP7	1.9382×10^{-2} (6.00×10^{-2}) −	1.9844×10^{-1} (6.95×10^{-2}) =	7.4225×10^{-3} (4.07×10^{-2}) −	2.1168×10^{-1} (5.83×10^{-2})
LIRCMOP8	$0.0000 \times 10^{+0}$ ($0.00 \times 10^{+0}$) −	1.0965×10^{-1} (1.04×10^{-1}) =	6.0278×10^{-3} (3.30×10^{-2}) −	1.3642×10^{-1} (9.86×10^{-2})
LIRCMOP9	9.6729×10^{-2} (2.87×10^{-2}) −	2.8800×10^{-1} (5.17×10^{-2}) =	9.7909×10^{-2} (2.76×10^{-2}) −	2.5944×10^{-1} (7.32×10^{-2})
LIRCMOP10	6.5082×10^{-2} (2.83×10^{-2}) −	2.4906×10^{-1} (1.24×10^{-1}) =	5.9470×10^{-2} (1.97×10^{-2}) −	1.9340×10^{-1} (1.40×10^{-1})
LIRCMOP11	1.3892×10^{-1} (4.07×10^{-2}) −	3.7183×10^{-1} (1.11×10^{-1}) +	1.6531×10^{-1} (3.18×10^{-2}) −	2.7786×10^{-1} (8.43×10^{-2})
LIRCMOP12	2.3159×10^{-1} (7.98×10^{-2}) −	3.7287×10^{-1} (7.52×10^{-2}) =	1.9288×10^{-1} (7.46×10^{-2}) −	4.0423×10^{-1} (7.72×10^{-2})
LIRCMOP13	3.3380×10^{-4} (1.46×10^{-4}) +	1.0577×10^{-2} (5.75×10^{-2}) +	5.3026×10^{-5} (9.73×10^{-5}) +	$0.0000 \times 10^{+0}$ ($0.00 \times 10^{+0}$)
LIRCMOP14	7.9656×10^{-4} (2.55×10^{-4}) +	2.6131×10^{-2} (7.93×10^{-2}) +	1.8352×10^{-4} (2.95×10^{-4}) +	1.7358×10^{-5} (5.94×10^{-5})
MW1	Infeasible	Infeasible	Infeasible	1.4722×10^{-1} (1.30×10^{-1})
MW2	3.0609×10^{-1} (1.50×10^{-1}) −	3.0403×10^{-1} (7.47×10^{-2}) −	3.2396×10^{-1} (1.49×10^{-1}) =	3.8124×10^{-1} (1.29×10^{-1})
MW3	2.0367×10^{-1} (2.06×10^{-1}) −	4.8768×10^{-1} (1.93×10^{-2}) −	2.2971×10^{-1} (1.71×10^{-1}) −	5.0566×10^{-1} (1.43×10^{-2})
MW4	Infeasible	3.1094×10^{-1}($0.00 \times 10^{+0}$) =	Infeasible	4.3348×10^{-1} (5.15×10^{-2})
MW5	Infeasible	9.8648×10^{-3} (1.56×10^{-2}) −	Infeasible	1.1962×10^{-1} (7.62×10^{-2})
MW6	2.2796×10^{-1} (5.47×10^{-2}) −	2.2878×10^{-1} (4.78×10^{-2}) −	2.0492×10^{-1} (6.42×10^{-2}) −	2.9226×10^{-1} (1.89×10^{-2})
MW7	3.7160×10^{-1} (6.69×10^{-2}) =	4.0684×10^{-1} (1.45×10^{-3})+	3.8315×10^{-1} (5.65×10^{-2}) −	4.0304×10^{-1} (1.59×10^{-3})
MW8	4.6429×10^{-1} (7.49×10^{-2}) =	4.0464×10^{-1} (4.27×10^{-2}) −	4.3323×10^{-1} (6.61×10^{-2}) −	4.8872×10^{-1} (2.38×10^{-2})
MW9	Infeasible	2.8487×10^{-1} (1.60×10^{-2}) =	Infeasible	1.3763×10^{-1} (1.70×10^{-1})
MW10	2.0985×10^{-1} (1.03×10^{-1}) =	8.2714×10^{-2} (1.02×10^{-2}) −	1.9919×10^{-1} (1.03×10^{-1}) =	2.2475×10^{-1} (9.60×10^{-2})
MW11	2.4123×10^{-1} (3.84×10^{-2}) −	4.4117×10^{-1} (1.10×10^{-3})+	2.3732×10^{-1} (3.80×10^{-2}) −	4.3830×10^{-1} (1.25×10^{-3})
MW12	Infeasible	1.3176×10^{-1} (1.04×10^{-1}) =	Infeasible	1.4185×10^{-1} (2.45×10^{-1})
MW13	1.9544×10^{-1} (7.54×10^{-2}) −	7.7809×10^{-2} (9.91×10^{-2}) −	1.9767×10^{-1} (7.52×10^{-2}) −	2.9592×10^{-1} (8.84×10^{-2})
MW14	4.7499×10^{-2} (2.15×10^{-2}) −	2.3013×10^{-2} (1.94×10^{-3}) −	3.3691×10^{-2} (1.39×10^{-2}) −	1.1746×10^{-1} (4.12×10^{-2})
+/−/=	2/23/3	5/13/14	2/24/2	

The feasible regions of the LIR-CMOP test suite are small. For some test problems, the constrained Pareto front exhibits sparse points, continuous segments, and discontinuous segments. A portion of the feasible region has closed, non-closed, and three-dimensional grid shapes. For the two-dimensional test problems, such as LIR-CMOP1 to LIR-CMOP12, IDCNSGA-III exhibits a clear advantage over the CMOEA-MS, BiCo, AGEMOEA-II, and NSGA-II algorithms in the comparison. There is no clear advantage over DSPCMDE in the comparison. Particularly in the two three-dimensional problems, LIR-CMOP13 to LIR-CMOP14, the performance is not satisfactory. This could be due to the fact that when the algorithm executes a certain operator and receives a higher reward, its inherent greediness favors continuously executing a single operator, leading to premature convergence of the population. Moreover, in high-dimensional problems, which have more local optima, the greediness issue of the algorithm becomes more pronounced.

For the MW test problems, the constraints have various forms. In general, the feasible region is very small, and it is divided from a very large infeasible region. Hence, the constrained Pareto front includes multiple isolated solutions. Overall, IDCNSGA-III exhibits a relatively significant advantage compared with the CMOEA-MS, AGEMOEAII, DSPCMDE, and NSGA-II algorithms. In comparison with BiCo, IDCNSGA-III shows a slight performance deficiency when facing problems with non-contiguous, narrow, and scattered constraint shapes such as MW2, MW6, and MW10. This may happen due to the dependence on well-defined infeasible solutions to provide the information guiding population evolution. The scattered, narrow, and non-contiguous nature of the constraint regions may lead to insufficient feedback information, thus limiting its regulatory effect.

In addition, a Friedman test is used to rank all the algorithms. As the ranking results presented in Table 4, IDCNSGA-III shows the best ranking, and shows significant advantages in comparison with other algorithms.

Table 4. Average rankings of the algorithms with Friedman test.

Algorithm	Ranking
IDCNSGA-III	3.8088
DSPCMDE	4.4412
CMOEA-MS	5.7794
BiCo	6.0588
AGEMOEA-II	6.3676
NSGA-II	6.5441

In summary, when compared with other algorithms, the proposed method in this paper has the best performance. This further elucidates the effectiveness of the proposed approach. In terms of the problem dimension, the dimensions of TREE problems are higher than 100, where IDCNSGA-III has more significant advantages than other algorithms. The possible explanation is that combining the ε constrained method and the multi-operator method can fully leverage the advantages of both methods. Loosening the constraints first and gradually tightening them can effectively reduce the search space and improve search efficiency. In addition, the multi-operator method can search the solution space more comprehensively and improve search quality. The comprehensive application of the two methods can better balance search efficiency and search quality, thereby exhibiting better performance in solving high-dimensional problems.

4.5.2. Compared with ε-Feasibility-Based Constrained Optimization Algorithms

Because our algorithm uses the ε-feasibility-based population state to guide the evolution, we need to compare other ε-feasibility-based constrained optimization algorithms to show the advantages. The selected state-of-the-art competitors include Trip, DPPPS, and CAEAD. For the test problems, as well as TREE, LIRCMOP, and MW, we also selected the RWMOP test suite to deeply test the performance of all algorithms. These problems are classified into five parts according to their domain: mechanical design problems; chemical engineering problems; process design and synthesis problems; power electronics problems; and power system problems. According to Kumar et al.'s analysis, the RWMOP test function contains various questions with different difficulty levels. Although the RWMOP benchmark suite contains testing issues with relatively low dimensions (up to 34), most problems are hard to solve with the most advanced algorithms. IDCNSGA-III and the selected comparative algorithms were independently run 30 times, obtaining HV indicator values. The performance statistics of HV are presented in Table 5, where the best results are highlighted, and the results for all algorithms that cannot find feasible solutions are removed. All the problems for which no single algorithm can find a feasible solution are removed from Table 5.

The experimental results illustrate that IDCNSGA-III has significant advantages in comparison with competitors. For problems TREE3 to LIRCMOP6, and MW9-MW14, IDCNSGA-III has the most prominent performance. Specifically, IDCNSGA-III achieved "34+/14−/7=", "36+/14−/5=" and "37+/13−/5=" on all test problems. In each test suite, for all existing problems IDCNSGA-III can find feasible solutions, while competitors cannot make it. Particularly in the MW test suite, there were five problems that competitors could not solve even one of. This indicates that our proposed algorithm is efficient and has promising performance in terms of applying ε-feasibility techniques.

Table 5. HV indicator statistics for IDNSGA-III and comparison algorithms.

Problem	TriP	DPPPS	CAEAD	IDCNSGA-III
RWMOP1	5.9308×10^{-1} (2.84×10^{-3}) −	5.9372×10^{-1} (2.23×10^{-3}) −	6.0515×10^{-1} (1.38×10^{-3}) =	6.0529×10^{-1} (8.02×10^{-4})
RWMOP2	2.5903×10^{-1} (1.18×10^{-1}) −	2.3218×10^{-1} (9.43×10^{-2}) −	3.6619×10^{-1} (1.41×10^{-2}) −	3.9251×10^{-1} (7.04×10^{-4})
RWMOP3	9.0001×10^{-1} (5.74×10^{-4}) +	8.9997×10^{-1} (5.96×10^{-4}) +	8.9798×10^{-1} (1.18×10^{-3}) +	8.9312×10^{-1} (1.65×10^{-3})
RWMOP4	8.5421×10^{-1} (3.76×10^{-3}) −	8.5443×10^{-1} (3.07×10^{-3}) −	8.5315×10^{-1} (3.39×10^{-3}) −	8.5873×10^{-1} (1.41×10^{-3})
RWMOP5	4.3215×10^{-1} (8.51×10^{-4}) +	4.3186×10^{-1} (1.17×10^{-3}) +	4.3341×10^{-1} (3.27×10^{-4}) +	3.4774×10^{-1} (5.79×10^{-2})
RWMOP6	2.7559×10^{-1} (3.70×10^{-4}) −	2.7536×10^{-1} (7.04×10^{-4}) −	2.7448×10^{-1} (1.06×10^{-3}) −	2.7585×10^{-1} (2.07×10^{-3})
RWMOP7	4.8340×10^{-1} (1.53×10^{-4}) =	4.8319×10^{-1} (2.54×10^{-4}) −	4.8398×10^{-1} (2.39×10^{-4}) +	4.8340×10^{-1} (6.23×10^{-4})
RWMOP8	2.5752×10^{-2} (1.95×10^{-4}) +	2.5758×10^{-2} (1.98×10^{-4}) +	2.5913×10^{-2} (8.24×10^{-5}) +	2.5611×10^{-2} (2.82×10^{-4})
RWMOP9	3.9371×10^{-1} (9.07×10^{-3}) −	3.9425×10^{-1} (7.85×10^{-3}) −	4.0953×10^{-1} (1.10×10^{-4}) −	4.0970×10^{-1} (1.01×10^{-4})
RWMOP10	8.4260×10^{-1} (2.08×10^{-3}) +	8.4157×10^{-1} (1.52×10^{-3}) +	8.4204×10^{-1} (1.91×10^{-3}) +	8.1216×10^{-1} (8.48×10^{-3})
RWMOP11	9.3081×10^{-2} (1.38×10^{-3}) −	9.2283×10^{-2} (1.15×10^{-3}) −	9.1840×10^{-2} (2.02×10^{-3}) −	9.4492×10^{-2} (8.63×10^{-4})
RWMOP12	5.5804×10^{-1} (1.29×10^{-3}) =	5.5752×10^{-1} (1.85×10^{-3}) −	5.5619×10^{-1} (1.81×10^{-3}) −	5.5854×10^{-1} (1.34×10^{-3})
RWMOP13	8.7506×10^{-2} (2.66×10^{-4}) −	8.7487×10^{-2} (2.89×10^{-4}) −	8.6793×10^{-2} (4.26×10^{-4}) −	8.7645×10^{-2} (1.40×10^{-4})
RWMOP14	6.1141×10^{-1} (4.00×10^{-3}) −	6.1217×10^{-1} (3.50×10^{-3}) −	6.1320×10^{-1} (1.23×10^{-3}) −	6.1683×10^{-1} (4.09×10^{-4})
RWMOP15	5.3143×10^{-1} (3.35×10^{-3}) −	5.2874×10^{-1} (5.41×10^{-3}) −	5.4037×10^{-1} (6.61×10^{-4}) =	5.4029×10^{-1} (5.74×10^{-4})
RWMOP16	7.6265×10^{-1} (2.48×10^{-4}) −	7.6259×10^{-1} (2.49×10^{-4}) −	7.6250×10^{-1} (2.70×10^{-4}) −	7.6316×10^{-1} (2.82×10^{-5})
RWMOP17	2.2013×10^{-1} (4.95×10^{-2}) =	2.2862×10^{-1} (3.68×10^{-2}) −	2.4094×10^{-1} (2.48×10^{-2}) =	2.4837×10^{-1} (2.04×10^{-2})
RWMOP18	4.0510×10^{-2} (6.33×10^{-6}) +	4.0509×10^{-2} (6.77×10^{-6}) +	4.0505×10^{-2} (4.03×10^{-6}) +	4.0481×10^{-2} (2.63×10^{-5})
RWMOP19	2.5834×10^{-1} (2.00×10^{-2}) −	2.5561×10^{-1} (2.33×10^{-2}) −	3.4259×10^{-1} (7.91×10^{-3}) −	3.5698×10^{-1} (3.83×10^{-3})
RWMOP21	3.1570×10^{-2} (8.25×10^{-5}) +	3.1621×10^{-2} (5.17×10^{-5}) +	3.1760×10^{-2} (8.09×10^{-7}) +	3.1508×10^{-2} (4.65×10^{-4})
RWMOP22	infeasible	infeasible	8.0800×10^{-1} (2.15×10^{-1})	infeasible
RWMOP23	9.9856×10^{-1} (4.53×10^{-16}) =	9.9856×10^{-1} $(0.00 \times 10^{+0})$ =	$1.0187 \times 10^{+0}$ (1.15×10^{-1}) =	$1.0510 \times 10^{+0}$ (1.21×10^{-1})
RWMOP25	2.4150×10^{-1} (2.47×10^{-5}) +	2.4149×10^{-1} (2.37×10^{-5}) +	2.4149×10^{-1} (1.95×10^{-5}) +	2.4136×10^{-1} (8.03×10^{-5})
RWMOP26	1.2680×10^{-1} (1.78×10^{-2}) −	1.2454×10^{-1} (1.62×10^{-2}) −	1.4953×10^{-1} (9.48×10^{-3}) =	1.4490×10^{-1} (6.30×10^{-3})
RWMOP27	$1.3860 \times 10^{+8}$ $(4.40 \times 10^{+8})$ −	$1.2481 \times 10^{+8}$ $(4.08 \times 10^{+8})$ =	$1.2597 \times 10^{+9}$ $(5.96 \times 10^{+9})$ =	$4.0381 \times 10^{+8}$ $(1.79 \times 10^{+9})$
RWMOP28	infeasible	infeasible	infeasible	3.4356×10^{-2} (1.09×10^{-2})
RWMOP29	7.5012×10^{-1} (1.71×10^{-2}) +	7.4954×10^{-1} (2.02×10^{-2}) −	7.7187×10^{-1} (7.56×10^{-3}) =	7.4998×10^{-1} (6.20×10^{-2})
TREE1	7.1186×10^{-1} (9.69×10^{-3}) −	7.4386×10^{-1} (8.51×10^{-3}) +	7.3571×10^{-1} (6.27×10^{-3}) =	7.2952×10^{-1} (1.64×10^{-2})
TREE2	7.5052×10^{-1} (7.85×10^{-3}) −	7.7012×10^{-1} (7.14×10^{-3}) +	7.6484×10^{-1} (4.35×10^{-3}) =	7.6076×10^{-1} (1.01×10^{-2})
TREE3	7.2121×10^{-1} (4.69×10^{-2}) −	7.5576×10^{-1} (1.42×10^{-2}) −	7.1745×10^{-1} (1.63×10^{-2}) −	7.9388×10^{-1} (9.08×10^{-3})
TREE4	5.8872×10^{-1} (6.74×10^{-2}) −	6.0943×10^{-1} (2.95×10^{-2}) −	4.7716×10^{-1} (5.14×10^{-2}) −	7.0300×10^{-1} (3.38×10^{-2})
TREE5	7.2876×10^{-1} (5.40×10^{-2}) −	7.2453×10^{-1} (2.13×10^{-2}) −	6.3036×10^{-1} (3.01×10^{-2}) −	7.8099×10^{-1} (2.67×10^{-2})
LIRCMOP1	1.0835×10^{-1} (1.46×10^{-2}) −	1.0634×10^{-1} (1.33×10^{-2}) −	1.0751×10^{-1} (2.72×10^{-2}) −	1.2444×10^{-1} (2.56×10^{-2})
LIRCMOP2	2.2951×10^{-1} (1.86×10^{-2}) −	2.2789×10^{-1} (1.92×10^{-2}) −	2.3089×10^{-1} (3.32×10^{-2}) −	2.4912×10^{-1} (2.00×10^{-2})
LIRCMOP3	9.9237×10^{-2} (1.21×10^{-2}) −	1.0128×10^{-1} (1.02×10^{-2}) −	8.9164×10^{-2} (1.50×10^{-2}) −	1.0955×10^{-1} (2.05×10^{-2})
LIRCMOP4	1.8951×10^{-1} (1.70×10^{-2}) −	1.8867×10^{-1} (1.62×10^{-2}) −	1.8284×10^{-1} (3.20×10^{-2}) −	2.0854×10^{-1} (1.79×10^{-2})
LIRCMOP5	2.8560×10^{-1} (6.95×10^{-3}) −	2.7503×10^{-1} (1.98×10^{-2}) −	2.6212×10^{-1} (5.20×10^{-2}) −	2.8561×10^{-1} (1.65×10^{-3})
LIRCMOP6	1.8882×10^{-1} (2.17×10^{-2}) −	1.8259×10^{-1} (1.95×10^{-2}) =	1.6545×10^{-1} (2.05×10^{-2}) −	1.9183×10^{-1} (1.32×10^{-3})
LIRCMOP7	2.1642×10^{-1} (1.13×10^{-2}) +	1.4293×10^{-1} (7.74×10^{-2}) −	4.2776×10^{-3} (2.34×10^{-2}) −	2.1168×10^{-1} (5.83×10^{-2})
LIRCMOP8	2.0367×10^{-1} (1.07×10^{-2}) =	4.6456×10^{-2} (7.33×10^{-2}) −	$0.0000 \times 10^{+0}$ $(0.00 \times 10^{+0})$ −	1.3642×10^{-1} (9.86×10^{-2})
LIRCMOP9	1.6996×10^{-1} (7.00×10^{-2}) −	1.7749×10^{-1} (1.06×10^{-1}) −	2.5966×10^{-1} (6.40×10^{-2}) =	2.5944×10^{-1} (7.32×10^{-2})
LIRCMOP10	8.4000×10^{-2} (2.52×10^{-2}) −	6.2384×10^{-2} (2.41×10^{-2}) −	1.3664×10^{-1} (7.23×10^{-2}) =	1.9340×10^{-1} (1.40×10^{-1})
LIRCMOP11	1.9344×10^{-1} (3.97×10^{-2}) −	1.5943×10^{-1} (6.25×10^{-2}) −	2.0644×10^{-1} (1.02×10^{-1}) −	2.7786×10^{-1} (8.43×10^{-2})
LIRCMOP12	3.4834×10^{-1} (5.79×10^{-2}) −	3.2052×10^{-1} (6.36×10^{-2}) −	2.7427×10^{-1} (6.43×10^{-2}) −	4.0423×10^{-1} (7.72×10^{-2})
LIRCMOP13	5.4827×10^{-2} (7.41×10^{-2}) +	2.6023×10^{-2} (5.53×10^{-2}) +	3.5855×10^{-5} (8.81×10^{-5}) +	$0.0000 \times 10^{+0}$ $(0.00 \times 10^{+0})$
LIRCMOP14	5.1190×10^{-2} (7.33×10^{-2}) +	2.2962×10^{-2} (4.48×10^{-2}) +	2.2361×10^{-4} (2.69×10^{-4}) +	1.7358×10^{-5} (5.94×10^{-5})
MW1	infeasible	infeasible	infeasible	1.4722×10^{-1} (1.30×10^{-1})
MW2	4.2862×10^{-1} (5.85×10^{-2}) =	4.3252×10^{-1} (5.50×10^{-2}) =	infeasible	3.8124×10^{-1} (1.29×10^{-1})
MW3	4.2942×10^{-1} (1.84×10^{-2}) −	4.2029×10^{-1} (2.10×10^{-2}) −	1.3968×10^{-1} $(0.00 \times 10^{+0})$ =	5.0566×10^{-1} (1.43×10^{-2})
MW4	infeasible	infeasible	infeasible	4.3348×10^{-1} (5.15×10^{-2})
MW5	infeasible	infeasible	infeasible	1.1962×10^{-1} (7.62×10^{-2})
MW6	3.1386×10^{-1} (1.20×10^{-2}) +	3.1343×10^{-1} (1.26×10^{-2}) +	8.1863×10^{-2} (4.78×10^{-2}) −	2.9226×10^{-1} (1.89×10^{-2})
MW7	4.0831×10^{-1} (9.17×10^{-4}) +	4.0742×10^{-1} (1.21×10^{-3}) +	4.0216×10^{-1} (4.37×10^{-3}) =	4.0304×10^{-1} (1.59×10^{-3})
MW8	5.2629×10^{-1} (9.09×10^{-3}) +	5.2888×10^{-1} (9.52×10^{-3}) +	3.1607×10^{-1} (7.02×10^{-2}) −	4.8872×10^{-1} (2.38×10^{-2})
MW9	infeasible	infeasible	infeasible	1.3763×10^{-1} (1.70×10^{-1})
MW10	1.6516×10^{-1} (6.00×10^{-2}) −	1.4508×10^{-1} (5.25×10^{-2}) −	infeasible	2.2475×10^{-1} (9.60×10^{-2})
MW11	4.3417×10^{-1} (4.16×10^{-3}) −	4.0024×10^{-1} (4.32×10^{-2}) −	infeasible	4.3830×10^{-1} (1.25×10^{-3})
MW12	infeasible	infeasible	infeasible	1.4185×10^{-1} (2.45×10^{-1})
MW13	2.0227×10^{-1} (7.81×10^{-2}) −	2.1554×10^{-1} (7.95×10^{-2}) −	$0.0000 \times 10^{+0}$ $(0.00 \times 10^{+0})$ −	2.9592×10^{-1} (8.84×10^{-2})
MW14	1.7216×10^{-2} (2.06×10^{-3}) −	1.6526×10^{-2} (1.52×10^{-3}) −	1.3666×10^{-2} (9.90×10^{-4}) −	1.1746×10^{-1} (4.12×10^{-2})
+/−/=	14/36/7	14/37/5	13/33/5	

4.5.3. Sensitivity Analysis of Key Parameters

To better set the parameters in our algorithm, the influence of key parameters on algorithm performance are tested with different parameters setting. α, γ, and ϵ control the update of current reward to Q-table, prediction ability of reward, and ϵ greedy, respectively.

Hence, we set the value of α with 0.001, 0.005, 0.01, 0.015, and 0.02, respectively. γ changes from 0.1 to 0.5 with an interval of 0.1, ϵ changes from 0.5 to 0.9 with an interval of 0.1. Each version of IDCNSGA-III is run 30 times on MW, LIRCMOP, and TREE test suites independently. All versions of IDCNSGA-III are compared with each other using Friedman test, and the ranking results of IDCNSGA-III on α, γ, and ϵ are presented in Tables 6–8, respectively. For α, $\alpha = 0.005$ has the best ranking, while $\alpha = 0.001$ and $\alpha = 0.01$ have similar performance. That means $\alpha = 0.005$ is a promising setting. For γ, a relatively small setting is better. $\gamma = 0.2$ shows the best performance than $\gamma = 0.3$, 0.4, and 0.5. For ϵ, $\epsilon = 0.8$ is a good choice, which surpasses $\epsilon = 0.5$, 0.6, 0.7, and 0.9.

Table 6. Average rankings of the algorithms with different α value (Friedman).

Algorithm	Ranking
IDCNSGA-III ($\alpha = 0.005$)	2.6029
IDCNSGA-III ($\alpha = 0.02$)	3.0294
IDCNSGA-III ($\alpha = 0.01$)	3.0735
IDCNSGA-III ($\alpha = 0.015$)	3.1471
IDCNSGA-III ($\alpha = 0.001$)	3.1471

Table 7. Average rankings of the algorithms with different γ value (Friedman).

Algorithm	Ranking
IDCNSGA-III ($\gamma = 0.2$)	2.8382
IDCNSGA-III ($\gamma = 0.3$)	2.9118
IDCNSGA-III ($\gamma = 0.4$)	3.0147
IDCNSGA-III ($\gamma = 0.5$)	3.1471
IDCNSGA-III ($\gamma = 0.1$)	3.2059

Table 8. Average rankings of the algorithms with different ϵ value (Friedman).

Algorithm	Ranking
IDCNSGA-III ($\epsilon = 0.8$)	2.6618
IDCNSGA-III ($\epsilon = 0.5$)	2.8382
IDCNSGA-III ($\epsilon = 0.9$)	3.0294
IDCNSGA-III ($\epsilon = 0.6$)	3.1618
IDCNSGA-III ($\epsilon = 0.7$)	3.3088

5. Conclusions

We propose a population feasibility state guided autonomous evolutionary optimization method for handling constrained multi-objective optimization problems. Taking DCNSGA-III as an example, our proposed method significantly improves its performance on four test suites, demonstrating the effectiveness of our approach. In addition, the comparison of IDCNSGA-III with five state-of-the-art constrained multi-objective optimization algorithms and three ε-feasibility based constrained multi-objective optimization algorithms also demonstrates the effectiveness of our approach. The significant performance of the RWMOP benchmark suite illustrate the scalability of our algorithm in dealing with real-world problems.

However, our method does not significantly improve the performance of the embedded algorithm for test problems with high-dimensional objectives and non-closed constraint shapes. To further enhance performance, we will investigate the following issues in our

future work: (1) Designing new methods for describing population state. (2) Integrating new reproduction operator.

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

Funding: This research was funded by the National Natural Science Foundation of China with grant No. 62303465, the Shandong Provincial Natural Science Foundation with grant No. ZR2022LZH017.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Hu, H.; Sun, X.; Zeng, B.; Gong, D.; Zhang, Y. Enhanced evolutionary multi-objective optimization-based dispatch of coal mine integrated energy system with flexible load. *Appl. Energy* **2022**, *307*, 118130. [CrossRef]
2. Gong, D.; Xu, B.; Zhang, Y.; Guo, Y.; Yang, S. A Similarity-Based Cooperative Co-Evolutionary Algorithm for Dynamic Interval Multiobjective Optimization Problems. *IEEE Trans. Evol. Comput.* **2020**, *24*, 142–156. [CrossRef]
3. Zuo, M.; Dai, G.; Peng, L.; Wang, M.; Liu, Z.; Chen, C. A case learning-based differential evolution algorithm for global optimization of interplanetary trajectory design. *Appl. Soft Comput.* **2020**, *94*, 106451. [CrossRef]
4. Kumar, A.; Wu, G.H.; Ali, M.Z.; Mallipeddi, R.; Suganthan, P.N.; Das, S. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm Evol. Comput.* **2020**, *56*, 100693. [CrossRef]
5. Liang, J.; Ban, X.; Yu, K.; Qu, B.; Qiao, K.; Yue, C.; Chen, K.; Tan, K.C. A Survey on Evolutionary Constrained Multiobjective Optimization. *IEEE Trans. Evol. Comput.* **2023**, *27*, 201–221. [CrossRef]
6. Zuo, M.C.; Dai, G.M.; Peng, L.; Tang, Z.; Gong, D.W.; Wang, Q.X. A differential evolution algorithm with the guided movement for population and its application to interplanetary transfer trajectory design. *Eng. Appl. Artif. Intell.* **2022**, *110*, 104727. [CrossRef]
7. Zuo, M.C.; Dai, G.M.; Peng, L. A new mutation operator for differential evolution algorithm. *Soft Comput.* **2021**, *25*, 13595–13615. [CrossRef]
8. Zuo, M.C.; Dai, G.M.; Peng, L. Multi-agent genetic algorithm with controllable mutation probability utilizing back propagation neural network for global optimization of trajectory design. *Eng. Optim.* **2019**, *51*, 120–139. [CrossRef]
9. Jiao, L.C.; Luo, J.J.; Shang, R.H.; Liu, F. A modified objective function method with feasible-guiding strategy to solve constrained multi-objective optimization problems. *Appl. Soft Comput.* **2014**, *14*, 363–380. [CrossRef]
10. Maldonado, H.M.; Zapotecas-Martínez, S. A Dynamic Penalty Function within MOEA/D for Constrained Multi-objective Optimization Problems. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 28 June–1 July 2021; pp. 1470–1477.
11. Fan, Z.; Ruan, J.; Li, W.J.; You, Y.G.; Cai, X.Y.; Xu, Z.L.; Yang, Z.; Sun, F.Z.; Wang, Z.J.; Yuan, Y.T.; et al. A Learning Guided Parameter Setting for Constrained Multi-Objective Optimization. In Proceedings of the 2019 1st International Conference on Industrial Artificial Intelligence, Shenyang, China, 22–26 July 2019.
12. Morovati, V.; Pourkarimi, L. Extension of Zoutendijk method for solving constrained multiobjective optimization problems. *Eur. J. Oper. Res.* **2019**, *273*, 44–57. [CrossRef]
13. Schütze, O.; Alvarado, S.; Segura, C.; Landa, R. Gradient subspace approximation: A direct search method for memetic computing. *Soft Comput.* **2017**, *21*, 6331–6350. [CrossRef]
14. Long, Q. A constraint handling technique for constrained multi-objective genetic algorithm. *Swarm Evol. Comput.* **2014**, *15*, 66–79. [CrossRef]
15. Vieira, D.A.G.; Adriano, R.L.S.; Vasconcelos, J.A.; Krähenbühl, L. Treating constraints as objectives in multiobjective optimization problems using niched pareto genetic algorithm. *IEEE Trans. Magn.* **2004**, *40*, 1188–1191. [CrossRef]
16. Ming, F.; Gong, W.; Wang, L.; Gao, L. Constrained Multi-objective Optimization via Multitasking and Knowledge Transfer. In Proceedings of the IEEE Transactions on Evolutionary Computation, Online, 20 December 2022.
17. 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]
18. Takahama, T.; Sakai, S. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. *IEEE Congr. Evol. Comput.* **2006**, *3*, 2322–2327.
19. Runarsson, T.P.; Xin, Y. Stochastic ranking for constrained evolutionary optimization. *IEEE Trans. Evol. Comput.* **2000**, *4*, 284–294. [CrossRef]

20. Saha, A.; Ray, T. Equality Constrained Multi-Objective Optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Brisbane, QLD, Australia, 10–15 June 2012.
21. Hamida, S.B.; Schoenauer, M. ASCHEA: New results using adaptive segregational constraint handling. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02, Honolulu, HI, USA, 12–17 May 2002; pp. 884–889.
22. Zapotecas-Martínez, S.; Ponsich, A. Constraint Handling within MOEA/D Through an Additional Scalarizing Function. *Genet. Evol. Comput. Conf.* **2020**, *6*, 595–602.
23. Liu, Z.Z.; Wang, Y.; Wang, B.C.A. Indicator-Based Constrained Multiobjective Evolutionary Algorithms. *IEEE Trans. Syst. Man Cybern.-Syst.* **2021**, *51*, 5414–5426. [CrossRef]
24. 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]
25. Ying, W.Q.; Peng, D.X.; Xie, Y.H.; Wu, Y. An Annealing Stochastic Ranking Mechanism for Constrained Evolutionary Optimization. In Proceedings of the 2016 International Conference on Information System and Artificial Intelligence, Hong Kong, China, 24–26 June 2016; pp. 576–580.
26. Sabat, S.L.; Ali, L.; Udgata, S.K. *Stochastic Ranking Particle Swarm Optimization for Constrained Engineering Design Problems*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; 672p.
27. Li, L.; Li, G.P.; Chang, L. On self-adaptive stochastic ranking in decomposition many-objective evolutionary optimization. *Neurocomputing* **2022**, *489*, 547–557. [CrossRef]
28. Ying, W.Q.; He, W.P.; Huang, Y.X.; Wu, Y. An Adaptive Stochastic Ranking Mechanism in MOEA/D for Constrained Multi-objective Optimization. In Proceedings of the 2016 International Conference on Information System and Artificial Intelligence, Hong Kong, China, 24–26 June 2016; pp. 514–518.
29. Gu, Q.H.; Wang, Q.; Xiong, N.N.; Jiang, S.; Chen, L. Surrogate-assisted evolutionary algorithm for expensive constrained multi-objective discrete optimization problems. *Complex Intell. Syst.* **2022**, *8*, 2699–2718. [CrossRef]
30. Wang, H.D.; Jin, Y.C. A Random Forest-Assisted Evolutionary Algorithm for Data-Driven Constrained Multiobjective Combinatorial Optimization of Trauma Systems. *IEEE Trans. Cybern.* **2020**, *50*, 536–549. [CrossRef]
31. Li, B.D.; Tang, K.; Li, J.L.; Yao, X. Stochastic Ranking Algorithm for Many-Objective Optimization Based on Multiple Indicators. *IEEE Trans. Evol. Comput.* **2016**, *20*, 924–938. [CrossRef]
32. Chen, Y.; Yuan, X.P.; Cang, X.H. A new gradient stochastic ranking-based multi-indicator algorithm for many-objective optimization. *Soft Comput.* **2019**, *23*, 10911–10929. [CrossRef]
33. 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]
34. 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]
35. Liu, Y.J.; Li, X.; Hao, Q.J. A New Constrained Multi-objective Optimization Problems Algorithm Based on Group-sorting. *Proc. Genet. Evol. Comput. Conf. Companion* **2019**, *7*, 221–222.
36. He, C.; Cheng, R.; Tian, Y.; Zhang, X.Y.; Tan, K.C.; Jin, Y.C. Paired Offspring Generation for Constrained Large-Scale Multiobjective Optimization. *IEEE Trans. Evol. Comput.* **2021**, *25*, 448–462. [CrossRef]
37. Tian, Y.; Li, X.P.; Ma, H.P.; Zhang, X.Y.; Tan, K.C.; Jin, Y.C. Deep Reinforcement Learning Based Adaptive Operator Selection for Evolutionary Multi-Objective Optimization. *IEEE Trans. Emerg. Top. Comput. Intell.* **2023**, *7*, 1051–1064. [CrossRef]
38. Zuo, M.; Gong, D.; Wang, Y.; Ye, X.; Zeng, B.; Meng, F. Process Knowledge-guided Autonomous Evolutionary Optimization for Constrained Multiobjective Problems. *IEEE Trans. Evol. Comput.* **2023**, *28*, 193–207. [CrossRef]
39. Jiao, R.W.; Zeng, S.Y.; Li, C.H.; Yang, S.X.; Ong, Y.S. Handling Constrained Many-Objective Optimization Problems via Problem Transformation. *IEEE Trans. Cybern.* **2021**, *51*, 4834–4847. [CrossRef]
40. 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]
41. Kumar, A.; Wu, G.H.; Ali, M.Z.; Luo, Q.Z.; Mallipeddi, R.; Suganthan, P.N.; Das, S. A Benchmark-Suite of real-World constrained multi-objective optimization problems and some baseline results. *Swarm Evol. Comput.* **2021**, *67*, 100961. [CrossRef]
42. Liang, J.; Qiao, K.J.; Yu, K.J.; Qu, B.Y.; Yue, C.T.; Guo, W.F.; Wang, L. Utilizing the Relationship Between Unconstrained and Constrained Pareto Fronts for Constrained Multiobjective Optimization. *IEEE Trans. Cybern.* **2023**, *53*, 3873–3886. [CrossRef]
43. 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.* **2022**, *52*, 9559–9572. [CrossRef]
44. Liu, Z.Z.; Wang, B.C.; Tang, K. Handling Constrained Multiobjective Optimization Problems via Bidirectional Coevolution. *IEEE Trans. Cybern.* **2022**, *52*, 10163–10176. [CrossRef]
45. Panichella, A. An Improved Pareto Front Modeling Algorithm for Large-scale Many-Objective Optimization. *Proc. Genet. Evol. Comput. Conf.* **2022**, *7*, 565–573.
46. Yu, K.J.; Liang, J.; Qu, B.Y.; Luo, Y.; Yue, C.T. Dynamic Selection Preference-Assisted Constrained Multiobjective Differential Evolution. *IEEE Trans. Syst. Man Cybern.-Syst.* **2022**, *52*, 2954–2965. [CrossRef]

47. Ming, F.; Gong, W.Y.; Wang, L.; Lu, C. A tri-population based co-evolutionary framework for constrained multi-objective optimization problems. *Swarm Evol. Comput.* **2022**, *70*, 101055. [CrossRef]
48. Zou, J.; Sun, R.Q.; Yang, S.X.; Zheng, J.H. A dual-population algorithm based on alternative evolution and degeneration for solving constrained multi-objective optimization problems. *Inf. Sci.* **2021**, *579*, 89–102. [CrossRef]