Operations Research and Fuzziology 运筹与模糊学, 2025, 15(2), 313-327
Published Online April 2025 in Hans. https://www.hanspub.org/journal/orf
https://doi.org/10.12677/orf.2025.152086

Hans汉斯

基于惩罚－模糊逻辑控制器的双种群进化算法

张裕漩

广东工业大学数学与统计学院，广东 广州

收稿日期：2025年2月16日；录用日期：2025年3月27日；发布日期：2025年4月8日

摘　要

针对约束多目标优化问题的复杂性，本文提出了一种基于模糊惩罚的双种群进化算法。首先，设计了一种基于模糊逻辑控制器的动态约束惩罚机制，该机制通过模拟人类思维自适应地调整惩罚因子，有效平衡了种群在可行解与不可行解之间的探索。其次，针对不同种群的进化需求，设计了差异化的子代生成机制，以增强种群的探索和开发能力。最后，利用信息共享机制，使得不同种群间能够共享优质解和有效的约束信息，进一步提高解的质量。将所提出的算法与目前最先进的六种约束多目标优化算法进行实验对比。实验结果表明，所提出的算法在复杂约束环境下展现出较强的鲁棒性，尤其能够有效处理决策变量耦合的问题。

关键词

约束多目标优化，多目标优化，模糊逻辑控制器

Penalty Fuzzy Logic Controller-Based Dual Population Evolutionary Algorithm

Yuxuan Zhang

School of Mathematics and Statistics, Guangdong University of Technology, Guangzhou Guangdong

Received: Feb. 16th, 2025; accepted: Mar. 27th, 2025; published: Apr. 8th, 2025

Abstract

In response to the complexity of constrained multi-objective optimization problems, this paper proposes a dual population evolutionary algorithm based on fuzzy penalties. Firstly, a dynamic constraint penalty mechanism based on a fuzzy logic controller is designed. This mechanism adaptively adjusts the penalty factor by simulating human thought processes, effectively balancing the exploration of feasible and infeasible solutions within the population. Secondly, a differentiated offspring generation mechanism is designed to address the evolutionary needs of different populations,

文章引用: 张裕漩. 基于惩罚-模糊逻辑控制器的双种群进化算法[J]. 运筹与模糊学, 2025, 15(2): 313-327.
DOI: 10.12677/orf.2025.152086

enhancing the exploration and exploitation capabilities of the population. Finally, different populations can share high-quality solutions and effective constraint information by utilizing the information-sharing mechanism, thereby further improving the solution quality. The proposed algorithm is compared experimentally with six state-of-the-art constrained multi-objective optimization algorithms. The experimental results demonstrate that the proposed algorithm exhibits significant robustness in complex constrained environments, particularly in its ability to effectively handle the coupling of decision variables.

Keywords

Constrained Multi-Objective Optimization, Multi-Objective Optimization, Fuzzy Logic Controller

1. 引言

约束多目标优化问题(Constrained Multi-Objective Optimization Problems, CMOPs)广泛存在于物流管理、能源优化和投资决策等领域中[1]。在不失一般性前提下，CMOPs 定义如下：

$$\begin{aligned} \min \quad & F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T \\ s.t \quad & g_j(x) \le 0, j = 1, 2, \ldots, p, \\ & h_k(x) = 0, k = 1, 2, \ldots, q, \\ & x = [x_1, x_2, \ldots, x_d]^T \in \Omega \end{aligned} \tag{1}$$

其中，$F(x) \in R^m$ 为目标向量，m 为目标空间的维数。$g_j(x)$ 为第 j 个不等式约束，p 为不等式约束的个数。$h_k(x)$ 为第 k 个等式约束，q 为等式约束的个数。$x \in \Omega \subset R^d$ 为决策变量，Ω 为 d 维的决策空间。每个解 x 的约束违反值如公式(2)所示。

$$\Phi(x) = \sum_{j=1}^{p} \max(g_j(x), 0) + \sum_{k=1}^{q} |h_k(x)| \tag{2}$$

对于一个解 $x = [x_1, x_x, \cdots, x_d]^{\mathrm{T}}$，满足 $\Phi(x) = 0$ 时，则认为解 x 为可行解，否则为不可行解，满足所有约束条件的解集称为可行域。CMOPs 中通常包含多个互相冲突的目标，这导致解空间不再是单一的最优解，而是形成了约束帕累托前沿(Concentrated Pareto Front, CPF) [2]。罚函数法作为一种经典的约束处理技术，其核心思想是对不可行解施加惩罚[3]。具体来说，罚函数法通过引入惩罚项，将解的约束违反值加入到目标函数中，构造出一个新的目标函数，形式如下：

$$\min \quad F'(x) = \left(f_1'(x), f_2'(x), \cdots, f_m'(x)\right)^{\mathrm{T}} \tag{3}$$

其中

$$f_i'(x) = f_i(x) + \rho \cdot \Phi(x), \ i = 1, 2, \cdots, m \tag{4}$$

$f_i'(x)$ 为第 i 个目标函加上惩罚项后的新目标函数；$\Phi(x)$ 作为惩罚项；ρ 为惩罚因子，用于调节惩罚项对目标函数的影响。惩罚因子的设置是罚函数法的关键，主要有三类设置方法：固定罚函数法、动态罚函数法和自适应罚函数法。

固定罚函数法在优化过程中保持惩罚因子恒定[4]。这种方法简单易行，但在面对复杂约束问题时，难以平衡约束与目标，因此所得到解的质量往往不理想。动态罚函数法在优化过程中调整惩罚因子，常采用指数递增或线性递增的方式[5]，但其高度依赖于函数的设计和参数调整。若调整不当，可能导致性能下降。相比之下，自适应罚函数法能够自动调整惩罚因子，表现出更大的灵活性和适应性[6]。然而，自适应方法的参数设计较为复杂，且随着问题复杂性的增加，还可能引发调整滞后问题。

针对上述方法的不足，本文创新性地引入模糊控制思想，设计了惩罚－模糊逻辑控制器(Penalty Fuzzy Logic Controller, PFLC)，并提出了一种基于惩罚－模糊控制器的约束多目标优化算法(Penalty Fuzzy Logic Controller-based Algorithm for Constrained Multi-objective Optimization, PFLCA)。所提出的控制器能够将搜索过程中的约束信息动态映射为惩罚因子，避免过度依赖单一个体特征。所提出的算法由两个种群组成，其中一个种群基于所设计的控制器专注于探索有效的约束信息，另一种群则专注于目标优化。同时，为满足不同种群的优化需求，分别采用了针对性的进化算法。最后，应用种群信息共享机制，实现种群之间的协同进化，并将所提出的算法与当前最先进的算法进行实验对比。实验结果表明，所提出的算法能够有效处理大部分的复杂约束优化问题，尤其是决策变量耦合问题。

本文其余部分安排如下：第二章详细介绍所提出算法的技术细节；第三章进行实验研究，并对结果进行分析；第四章中对文章进行总结，并提出了对未来工作的建议。

2. 提出算法

2.1. 算法框架

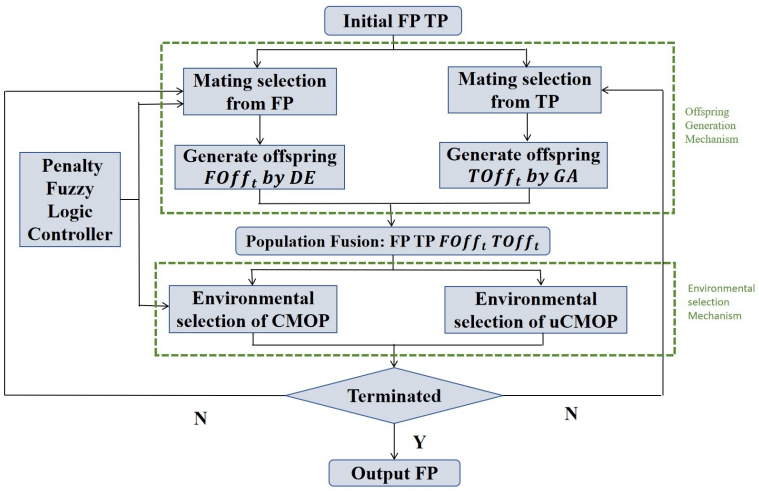


Figure 1. Framework diagram of the algorithm
图 1. 算法框架图

图 1 展示了所提出算法的整体框架。该算法设计了两个协同工作的种群：模糊种群(Fuzzy Population, FP)和目标种群(Target Population, TP)。FP 主要保障解的可行性，而 TP 则专注于目标函数的优化。两种群的协同合作确保解的多样性与全局分布的均衡性。控制器 PFLC 作用于模糊种群进化中的交配选择和环境选择过程，用于引导模糊种群生成更多有潜力可行解，并提升种群融合后的整体质量。同时，将生成后的子代与父代进行种群融合，实现信息共享。

算法 1 概述了所提出算法 PFLCA 的主要步骤。首先，确定算法的输入与输出，分别为种群规模 N 和最大进化代数 T，输出为 FP。在初始化阶段，生成初始化模糊种群 FP_0 和目标种群 TP_0 并初始化 PFLC。

其次，当未达到最大进化代数 T 时，调用 PFLC 计算惩罚因子 ρ_t，利用算法 2 生成 FP 和 TP 的子代。将种群 $FP_t \cup TP_t \cup FOff_t \cup TOff_t$ 和 $TP_t \cup TP_t \cup FOff_t \cup TOff_t$ 合并，实现信息共享。最后，根据算法 3 挑选下一代种群 FP_{t+1} 和 TP_{t+1}。在下面的段落中，将逐步解释 PFLCA 的细节。

算法 1：算法主要步骤

输入：种群规模 N，最大进化代数 T；

输出：模糊种群 FP；

1 初始化：生成初始种群 FP_0 和 TP_0、初始化 PFLC；

2 当 $t < T$ 时，执行下列步骤：

3 调用 PFLC 计算 ρ_t；

4 调用算法 2 生成子代种群 $FOff_t$、$TOff_t$；

5 合并种群，并根据算法 3 选择下一代种群 FP_{t+1} 和 TP_{t+1}；

6 $t = t + 1$.

2.2. 惩罚 - 模糊逻辑控制器

模糊逻辑控制器从输入到输出主要经过三个步骤[7]，其原理图如图 2 所示。首先，在模糊阶段将精确的输入值通过隶属度函数转换为模糊集合中的隶属度值，即模糊输入。其次，根据模糊规则库对模糊输入进行逻辑推理，生成模糊输出。最后为解模糊化，根据将模糊输出转化为精确的控制信号。

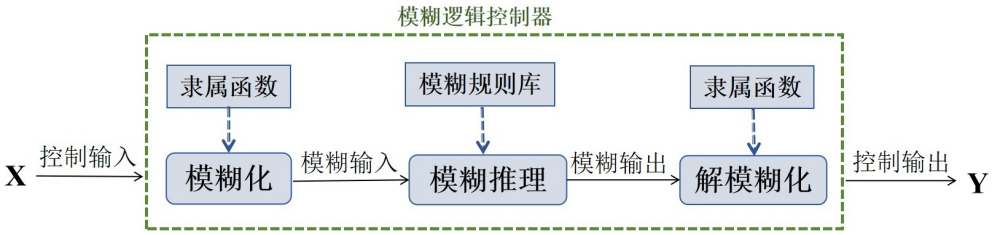


Figure 2. Schematic diagram of the fuzzy logic controller
图 2. 模糊逻辑控制器原理图

2.2.1. 输入与输出的定义

PFLC 的输入包含两部分，分别为约束违反均值 $\Phi_{avg}(t)$ 和可行解比例 $rf(t)$。输出为惩罚因子 ρ_t。

(1) $\Phi_{avg}(t)$：在第 t 代中，种群中 N 个解的约束违反值为 $\{\Phi_1(t), \Phi_2(t), \cdots, \Phi_N(t)\}$，计算其平均值作为控制器的第一个输入，定义如下：

$$\Phi_{avg}(t) = \frac{\sum_{i=1}^{N} \Phi_i(t)}{N} \tag{5}$$

其中，表示第 i 个解的约束违反值，N 为种群规模。

(2) $rf(t)$：第 t 代中，种群中可行解个数为 $N_f(t)$，种群总规模为 $N(t)$。$rf(t)$ 作为控制器的第二个输入，定义如下：

$$rf(t) = \frac{N_f(t)}{N(t)} \tag{6}$$

2.2.2. 动态论域的设计

为了提高控制器的适应性，PFLC 的论域为动态变化的。动态的论域设计可以根据系统状态的变化，增强控制的鲁棒性和控制精度。以下对输入、输出的动态论域设计进行详细说明。

(1) 约束违反均值 $\Phi_{avg}(t)$ 的论域 CVV 随种群进化动态调整，定义为：

$$\left[\mathrm{CVV}_d(t),\mathrm{CVV}_u(t)\right]=\left[\max\left(\Phi_{\min}(t),0\right),\min\left(\Phi_{\max}(t),\mathrm{CVV}_u(t-1)\right)\right] \tag{7}$$

其中 $\mathrm{POF}_d(t)$ 和 $\mathrm{POF}_u(t)$ 分别为第 t 代约束违反均值论域 CVV 的下限和上限。$\Phi_{\min}(t)$、$\Phi_{\max}(t)$ 为第 t 代解中的最小和最大约束违反值。

(2) 可行解比例 $rf(t)$ 的论域 POF 设计为固定区间：

$$\left[\mathrm{POF}_d(t),\mathrm{POF}_u(t)\right]=[0,1] \tag{8}$$

其中 $\mathrm{POF}_d(t)$ 和 $\mathrm{POF}_u(t)$ 分别为其上限和下限。由于 $rf(t)$ 在整个进化过程都在 $[0,1]$ 内，因此采取固定论域设计。

(3) 惩罚因子 ρ_t 的论域 $\sigma(t)$ 设计为动态变化，定义如下：

$$\left[\sigma_d(t),\sigma_u(t)\right]=\left[\max\left(\Phi_{\min}(t),\sigma_d(t-1)\right),3\right] \tag{9}$$

其中 $\sigma_d(t)$、$\sigma_u(t)$ 为第 t 代惩罚因子论域 $\sigma(t)$ 的上限和下限。下限固定为 3 能够有效避免惩罚因子过大或过小导致种群陷入局部最优或无法逼近 CPF。

2.2.3. 模糊集合与隶属度函数

对于每个输入，定义了 3 个模糊集，分别为负大(Negative Big, NB)、零(Zero, ZO)、正大(Positive Big, PB)。将输入论域分为 2 个相等的部分，具体如下：

$$\Delta_{\mathrm{CVV}}=\frac{\mathrm{CVV}_u(t)-\mathrm{CVV}_d(t)}{2} \tag{10}$$

$$\Delta_{\mathrm{POF}}=\frac{\mathrm{POF}_u(t)-\mathrm{POF}_d(t)}{2} \tag{11}$$

对于输出变量惩罚因子 ρ_t，定义了 5 个模糊集：负大(NB)、负小(Negative Small, NS)、零(ZO)、正小(Positive Small, PS)、正大(PB)。ρ_t 的模糊集设计更加细化，使其能够在推理阶段提供更丰富的决策空间。将输出论域分为 4 个相等的部分，如公式 12 所示。

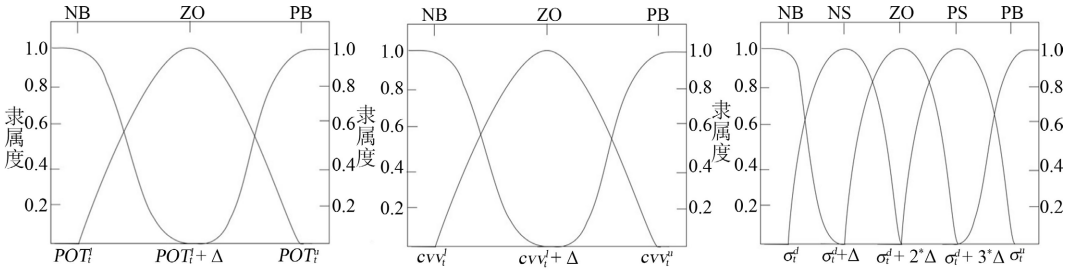
$$\Delta_{\sigma}=\frac{\sigma_u(t)-\sigma_d(t)}{4} \tag{12}$$

图 3 展示了所构建的输入与输出变量的隶属函数图。它们是由 S 型隶属函数(公式 13)、Z 型隶属函数(公式 14)及组合函数(公式 15)所构建的。

$$f_s(x;a,b)=\begin{cases}0 & x\le a\\ 2\left(\dfrac{x-a}{b-a}\right)^2 & a\le x\le\dfrac{a+b}{2}\\ 1-2\left(\dfrac{x-b}{b-a}\right)^2 & \dfrac{a+b}{2}\le x\le b\\ 1 & x\ge b\end{cases} \tag{13}$$

$$f_s(x;a,b)=\begin{cases}1 & x\le a\\ 1-2\left(\dfrac{x-a}{b-a}\right)^2 & a\le x\le \dfrac{a+b}{2}\\ 2\left(\dfrac{x-b}{b-a}\right)^2 & \dfrac{a+b}{2}\le x\le b\\ 0 & x\ge b\end{cases} \tag{14}$$

$$f_{SZ}(x;a,b,c)=\begin{cases}f_S(x;a,b) & x\le b\\ f_Z(x;a,b) & x>b\end{cases} \tag{15}$$



(a) POF模糊函数图　(b) CVV模糊函数图　(c) σ模糊函数图

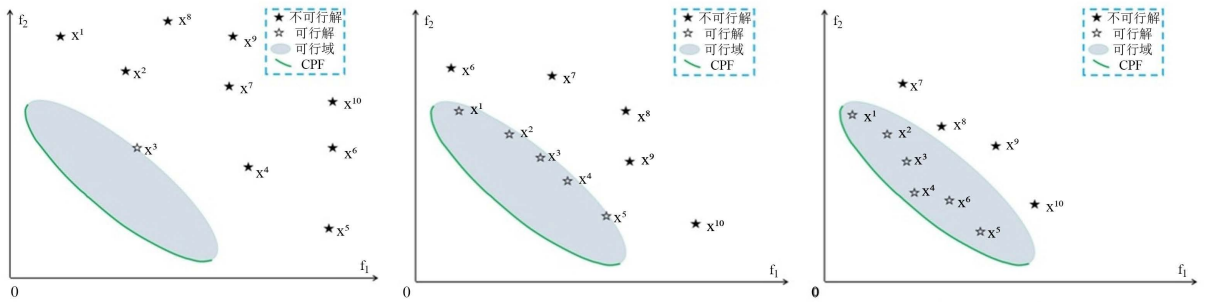
Figure 3. Graphs of membership functions for input and output variables
图 3. 输入与输出变量隶属函数图

2.2.4. 模糊规则库

在模糊规则库的构建过程中，本文遵循全面性、平滑性和鲁棒性原则。首先，基于罚函数思想预设多个规则库。然后，对预设定的规则库进行对比实验并调整优化。最后，根据实验结果选出最优的规则库。所构建的最终规则库如表 1 所示，下面以#号标记的模糊规则作为示例，进行详细说明。

Table 1. Fuzzy rule base
表 1. 模糊规则库

$\Phi_{avg}(t)$ \ $rf(t)$	NB	ZO	PB
NB	ZO	PS	PS#3
ZO	NS	ZO#2	PS
PB	NB#1	NS	ZO



(a)进化早期　(b)进化中期　(c)进化后期

Figure 4. Diagram of population evolution status
图 4. 种群进化状态示意图

- #1：若 $rf(t)$ 为 NB，$\Phi_{avg}(t)$ 为 PB，则 ρ_t 为 NB。

此情形多出现于种群进化早期，如图 4(a)所示，大部分解为不可行解且偏离可行域。采取较小的惩罚因子，能够使拥有较好目标信息的解得以保留，促使种群继续探索可行域。

- #2：若 $rf(t)$ 为 ZO，$\Phi_{avg}(t)$ 为 ZO，则 ρ_t 为 ZO。

此情形多出现于种群进化中期，如图 4(b)所示，已有部分解进入可行域且约束违反值明显下降。采取适中的惩罚因子，不仅能够引导不可行解向可行域迁移，同时继续探索潜在可行域。

- #3：若 $rf(t)$ 为 PB、$\Phi_{avg}(t)$ 为 NB，则 ρ_t 为 PS。

此情形多出现于种群进化后期，如图 4(c)所示，绝大部分解为可行解，并且不可行解位于可行域边界。采取较大的惩罚因子，能够促使不可行解快速进入可行域，从而迅速逼近 CPF。

2.2.5. 解模糊化

在解模糊阶段，本文采用面积质心法对推理结果进行解模糊。面积质心法通过计算模糊输出隶属度曲线的重心来确定最终输出值，其基本定义如下：

$$f=\frac{\sum_{k=1}^{|R|}y^k\cdot\mu_b^k\left(y^k\right)}{\sum_{k=1}^{|R|}\mu_b^k\left(y^k\right)} \tag{16}$$

其中 f 为去模糊化后的输出结果；$|R|$ 为模糊规则条数；μ_B^k 和 y^k 分别为隶属函数和第 k 条规则的输出结果。

2.3. 子代种群生成机制

在种群进化中，本文引用基于 Minkowski 无穷范数的比率指标，用于评价个体之间的相对优劣，能够优先识别并剔除异常解，对边界点具有天然的保护作用，其定义如下[8]：

$$I_p^r(u|v)=\begin{cases}\|R(u,v)\|_p & \text{若}\exists v_q>u_q\\ -\|R(u,v)\|_p & \text{其他情况}\end{cases} \tag{17}$$

其中

$$R(u,v)=\left(\max\left(0,\frac{v_1}{u_1}-1\right),\cdots,\max\left(0,\frac{v_m}{u_m}-1\right)\right) \tag{18}$$

$\|\cdot\|$ 表示范数。将解 x_i 与解集 X 中的所有解进行比较，选取最小值作为解 x_i 的指标。

算法 2 展示了生成子代种群的伪代码。模糊种群的子代生成机制包括三个主要步骤。首先，针对输入的 FP_t，计算每个个体的约束违反值，并利用惩罚因子 ρ_t 对个体施加惩罚，得到惩罚后的种群 FP_t'。其次，根据公式(17)计算惩罚后 FP_t' 中个体的指标，并将种群按照指标值降序排序。根据排序结果将种群划分为三组：第一组为前 1/3 的个体，即指标值较大的个体；第二组为后 1/3 的个体，即指标值较低的个体；第三组为剩余中间部分的个体。最后，基于分组结果应用 DE/best/1/bin 算子生成子代 $FOff_t$。对于目标种群，则根据公式(17)计算基于原目标函数的个体指标，再根据设定的概率随机选择配偶，再利用遗传算法产生子代 $TOff_t$。

在子代生成机制中，两个种群采用了不同的策略，该设计的核心思想是利用进化算子实现功能互补。模糊种群利用差分算子的高效全局搜索能力，能够有效扩展搜索范围，提高解的多样性。目标种群则利

用遗传算法的局部开发能力和计算效率优势，确保种群能够更快地逼近 CPF，提高解的收敛性。

算法 2：子代种群生成

输入：模糊种群 FP_t、惩罚因子 ρ_t、目标种群 TP_t；

输出：模糊种群子代 $FOff_t$、目标种群子代 $TOff_t$；

#模糊种群子代生成

1 计算 FP_t 中个体的约束违反值；

2 FP_t'：利用 ρ_t 对 FP_t 中的个体施加惩罚；

3 计算 FP_t' 中个体的指标；

4 根据指标值对个体进行分类；

5 应用差分算子生成模糊子代 $FOff_t$；

#目标种群子代生成

6 确认 TP_t 中每个个体的邻居个体；

7 根据随机概率选择个体作为配偶；

8 应用遗传算法生成目标子代 $TOff_t$

2.4. 种群选择

算法 3：种群选择

输入：当前种群 P_t；

输出：种群 P_{t+1}；

1 计算 P_t 中个体指标；

2 将 P_t 中指标值大于 0 的非支配解放入 P_{t+1}；

3 若 $|P_{t+1}|$ 小于 N，则保留指标较优的前 N 个个体；

4 若 $|P_{t+1}|$ 大于 N，则根据并行距离逐步移除指标较差的个体；

5 若 $|P_{t+1}|$ 等于 N，则输出 P_{t+1}。

算法 3 展示了种群选择的伪代码。首先，利用公式(17)计算个体的指标，评估个体的优劣，将优势个体放入 P_{t+1} 中。其次，根据已有的 P_{t+1} 中个体的个数，若个体不足，则进行个体的补充；若个体数超出种群个数 N，则利用并行距离的计算对指标值较差的个体进行筛减，维护种群在目标空间中的多样性，避免解过于集中。值得注意的是，种群的选择方法来源于 PREA，不仅保证了更新后种群能够拥有更好的收敛性，同时也显著提升了个体在解空间的分布均匀性[8]。

2.5. 计算复杂度

算法 PFLCA 的复杂度由指标计算、子代生成、种群选择和控制器 PFLC 决定。指标计算的复杂度为 $O\left(mN^2\right)$ [8]，其中 m 为目标个数，N 为种群规模。在子代生成中，差分算子和遗传算子都基于决策变量 D 进行操作，因此其复杂度为 $O\left(ND\right)$。在种群选择中需要更新部分解的指标，在最不理想的情况下，其复杂度为 $O\left(N^3\right)$。PFLC 需要遍历模糊种群的所有个体，其复杂度为 $O\left(N\right)$。综合所有阶段，PFLCA 的复杂度为 $O\left(mN^2+ND+N^3+N\right)$，进一步简化为 $O\left(N^3\right)$。

3. 实验分析

3.1. 对比算法与基准函数

本文选用了六种目前最先进的 CMOEA 作为对比算法：C3M [9]、CMOQLMT [10]、ICMA [11]、IM-CMOEA/D [12]、CCMO [13]和 t-DEA-CPBI [14]。对比算法的核心思想如下：

- C3M：基于分阶段思想的优化算法，根据 CPF 的不同，对约束条件进行等级分类，将优化过程分为目标优化，逐级约束优化以及全局优化三个阶段[9]。
- CMOQLMT：基于多任务框架的优化算法，利用 Q-Learning 技术，根据种群进化状态动态选择最适合的辅助任务，并且主任务与辅助任务间采用不同的约束处理技术[10]。
- ICMA：基于指标的优化算法，利用基于成本距离的约束指标对个体进行筛选[11]。
- IM-CMOEA/D：基于多种群进化思想的算法，其利用 K-means 聚类将种群划分为多个子种群，并使用逆向模型生成分布均匀的候选解[12]。
- CCMO：经典的协同优化算法，通过两个相互独立的种群分别处理 CMOPs 和针对目标的辅助问题[13]。
- t-DEA-CPBI：基于惩罚思想的优化算法，在传统 PBI 方法中引入约束违反值作为第三个优化指标[14]。

实验中选用了 LIR-CMOP [15]和 ZXH-CF [16]测试问题作为基准测试函数。ZXH-CF 系列问题的特点是决策变量不可分离，且 CPF 位于非线性曲面。LIR-CMOP 系列问题的特点是解空间中具有较大的不可行域。

3.2. 评价指标与参数设置

为了全面评估算法的有效性，本文采用三种性能指标：反向世代距离(Inverted Generational Distance, IGD)、超体积(Hypervolume, HV)和可行比例(Feasible Rate, FR)。这些指标分别评估解集的收敛性、多样性和可行性，全面反映算法的性能。IGD 表示解集与真实解之间的最小距离[17]；HV 表示解集与参考点所围成的超体积[18]；FR 表示解集中可行解的占比。IGD 值越小，HV 值和 FR 值越大，说明解集的质量越高。

为了确保公平性，各算法的参数和算子均按照文献中推荐进行设置，以保证其性能的最优发挥。所有测试问题的种群规模设为 100，最大评估次数为 15000，每个算法在测试问题上独立运行 30 次。实验结果均经过显著性水平为 0.05 的 Wilcoxon 秩和检验，并以表格形式展示。表格中对每个基准函数的最优值进行加粗显示，并在底部列出了 Wilcoxon 秩和检验的统计结果，符号“+”、“−”和“=”分别表示算法显著优于、劣于或统计学上与 PFLCA 持平。

3.3. 实验结果与分析

3.3.1. 算法性能对比分析

Table 2. Wilcoxon rank-sum test results for IGD, HV, and FR values obtained by all algorithms on 30 test problems
表 2. 所有算法在 30 组测试问题获得的 IGD、HV 和 FR 值的 Wilcoxon 秩和检验统计结果

算法	C3M	CMOQLMT	IMCMOEAD	ICMA	CCMO	tDEACPBI	PFLCA
IGD (+/−/=)	3/23/4	2/26/2	1/26/3	2/26/2	5/18/7	4/25/1	——
HV (+/−/=)	2/25/3	1/26/3	0/26/4	1/25/4	4/15/11	4/25/1	——
FR (+/−/=)	0/8/22	1/3/26	0/12/18	4/6/20	5/2/23	4/2/24	——

表 2 展示了所有算法在 30 组基准函数上的 IGD、HV 和 FR 值的 Wilcoxon 秩和检验统计结果，每个

基准函数的详细结果见附表 A1~A3。实验结果表明，算法 PFLCA 在大部分基准函数上展现出了显著优势，能够利用协同进化和 FLC 的优势维持解在多样性、收敛性和可行性之间的有效平衡，下面进行具体分析。

- 收敛性：PFLCA 在 ZXH-CF 和 LIR-CMOP 问题中分别获得了 10 个和 9 个 IGD 最优值。由表 2 的 IGD 统计结果可知，PFLCA 在 18 组基准函数优于 CCMO，并超过 25 组基准函数优于其他算法。实验结果表明，PFLCA 的解集更接近问题的真实解，表现出良好的收敛性。
- 多样性：PFLCA 在 ZXH-CF 和 LIR-CMOP 问题中分别获得了 11 个和 10 个 HV 最优值。由表 2 的 HV 统计结果可知，PFLCA 显著优于其他算法，尤其在 LIR-CMOP5-14 问题中表现突出，充分表明 PFLCA 能够保证解在 CPF 上均匀分布，尤其是在 CPF 位于解空间边界的问题上展现出良好的多样性。PFLCA 与 CCMO 均为双种群进化算法，但 PFLCA 在多样性表现上略优于 CCMO，这进一步验证了所提出的模糊逻辑控制器能够有效引导种群充分挖掘潜在可行域，从而提高算法的多样性。
- 可行性：PFLCA 在 ZXH-CF 和 LIR-CMOP 问题中分别获得了 12 个和 10 个 FR 最大值(即 FR = 1)。由表 2 的 FR 统计结果可知，PFLCA、CCMO 和 CMOQLMT 在可行性上表现相当，且显著高于其他算法，这说明 PFLCA 在求解过程中能够获得较高质量的可行解。

表 3 展示了所有算法在 30 组基准函数的平均运行时间及排名。实验结果表明，PFLCA 的平均运行时间仅为 1.1024 秒，排名第二，仅次于 t-DEACPBI。值得注意的是，PFLCA 在两个系列问题中相较于 t-DEACPBI 具有显著优势。这表明 PFLCA 在保持较优性能的同时，也能保证较高的计算效率。

Table 3. Average running time of all algorithms on 30 test problems
表 3. 所有算法在 30 个测试问题中的平均运行时间

算法	C3M	CMOQLMT	IMCMOEAD	ICMA	CCMO	tDEACPBI	PFLCA
平均运行时间(秒)	1.1788	1.3261	1.2113	2.3003	2.5037	0.9894	1.1024
排名	3	5	4	6	7	1	2

综合所有实验结果，C3M 通过分阶段搜索获得的解具有较高的可行性，CMOQLMT 基于多任务框架帮助种群准确逼近 CPF，t-DEA-CPBI 在求解效率上具有显著优势，但它们容易受到变量非线性干扰，从而陷入局部最优。ICMA 在 CPF 分段分布问题中发挥了指标计算的优势，但难以全面逼近曲面型 CPF。IM-CMOEA/D、CCMO 与 PFLCA 作为多种群优化算法，在 ZXH-CF 系列问题中，PFLCA 与 CCMO 能够有效地将解分布在非线性曲面上，而 IM-CMOEA/D 因依赖于约束处理机制的选择，导致在该系列问题中表现较弱。此外，在 LIR-CMOP 系列问题中，PFLCA 相比其他多种群算法的优势更为显著，充分证明了控制器能够引导种群突破不可行域，进一步验证了模糊控制器的核心规则库设计的有效性。

3.3.2. 双种群性能分析

为了进一步验证所提出的双种群框架的有效性，本文设计了四种算法变体：PFLCA-FP、PFLCA-TP、PFLCA-GA 和 PFLCA-DE。其中，PFLCA-FP 和 PFLCA-TP 分别为仅包含模糊种群和目标种群的变体。PFLCA-GA 和 PFLCA-DE 则分别将 PFLCA 的进化算子统一替换为遗传算子和差分算子。

表 4 展示了所有算法变体与 PFLCA 在 30 组测试问题上的 HV、IGD 和 FR 值的 Wilcoxon 秩和检验统计结果。实验结果表明，PFLCA 在所有性能指标上都显著优于各算法变体。具体来说，在 IGD 统计结果中，PFLCA 在 28 个基准函数上优于 PFLCA-FP；在 FR 统计结果中，PFLCA 在 27 个基准函数上优于 PFLCA-TP，这充分说明了双种群进化框架相比单种群在求解复杂问题的优势。由表 4 可知，PFLCA 在 IGD 统计结果中有 23 个基准函数优于 PFLCA-DE，在 HV 统计结果中有 22 个基准函数优于 PFLCA-GA。

这说明相比单一进化算子，差异化的算子设计能够增加算法的收敛性多样性，这也验证了第 2.3 小节中子代生成机制设计的正确性。

Table 4. Wilcoxon rank-sum test results for population performance
表 4. 种群性能的 Wilcoxon 秩和检验统计结果

算法	PFLCA-FP	PFLCA-TP	PFLCA-GA	PFLCA-DE	PFLCA
IGD (+/−/=)	0/28/2	1/23/6	4/11/15	2/23/5	——
HV (+/−/=)	2/24/4	1/24/5	2/22/6	5/13/12	——
FR (+/−/=)	4/7/19	0/27/3	3/8/19	2/8/20	——

4. 总结

针对罚函数法，本文设计了一种用于调整惩罚因子的模糊逻辑控制器。通过分析种群进化所有可能状态，构建了三输入五输出的模糊逻辑控制器。通过不断优化实验，构建了全面的规则库，实现了对优质解的深度挖掘。同时，将所提出的控制器与比率指标结合，提出了一个基于双种群进化思想的算法，该算法利用信息共享机制和差异化的进化算子，进一步提升了算法的求解性能。实验结果表明，本文提出的算法能够有效处理大部分约束多目标优化问题，尤其是具有较大不可行域和决策变量耦合的问题。尽管该控制器能够有效引导种群进化，从而提升算法求解效率，但规则库的设计依赖于领域知识和实验优化，在处理大规模优化问题时将增加其设计的复杂性。未来，我们将进一步探索基于数据驱动的方法，从实验数据中学习并生成模糊规则，并将算法扩展到大规模约束优化问题中。

参考文献

[1] Gu, F., Liu, H., Cheung, Y. and Liu, H. (2023) A Constrained Multiobjective Evolutionary Algorithm Based on Adaptive Constraint Regulation. *Knowledge-Based Systems*, **260**, Article ID: 110112. https://doi.org/10.1016/j.knosys.2022.110112

[2] Liang, J., Ban, X., Yu, K., Qu, B., Qiao, K., Yue, C., *et al*. (2023) A Survey on Evolutionary Constrained Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, **27**, 201-221. https://doi.org/10.1109/tevc.2022.3155533

[3] Yeniay, Ö. (2005) Penalty Function Methods for Constrained Optimization with Genetic Algorithms. *Mathematical and Computational Applications*, **10**, 45-56. https://doi.org/10.3390/mca10010045

[4] Jiao, L., Luo, J., Shang, R. and Liu, F. (2014) A Modified Objective Function Method with Feasible-Guiding Strategy to Solve Constrained Multi-Objective Optimization Problems. *Applied Soft Computing*, **14**, 363-380. https://doi.org/10.1016/j.asoc.2013.10.008

[5] Xia, Z., Liu, Y., Lu, J., Cao, J. and Rutkowski, L. (2021) Penalty Method for Constrained Distributed Quaternion-Variable Optimization. *IEEE Transactions on Cybernetics*, **51**, 5631-5636. https://doi.org/10.1109/tcyb.2020.3031687

[6] Ma, Z. and Wang, Y. (2023) Shift-based Penalty for Evolutionary Constrained Multiobjective Optimization and Its Application. *IEEE Transactions on Cybernetics*, **53**, 18-30. https://doi.org/10.1109/tcyb.2021.3069814

[7] Xue, F., Sanderson, A.C., Bonissone, P.P., and Graves, R.J. (2005) Fuzzy Logic Controlled Multi-Objective Differential Evolution. *The* 14*th IEEE International Conference on Fuzzy Systems*, Reno, 25 May 2005, 720-725.

[8] Yuan, J., Liu, H., Gu, F., Zhang, Q. and He, Z. (2021) Investigating the Properties of Indicators and an Evolutionary Many-Objective Algorithm Using Promising Regions. *IEEE Transactions on Evolutionary Computation*, **25**, 75-86. https://doi.org/10.1109/tevc.2020.2999100

[9] Sun, R., Zou, J., Liu, Y., Yang, S. and Zheng, J. (2023) A Multistage Algorithm for Solving Multiobjective Optimization Problems with Multiconstraints. *IEEE Transactions on Evolutionary Computation*, **27**, 1207-1219. https://doi.org/10.1109/tevc.2022.3224600

[10] Ming, F., Gong, W. and Gao, L. (2023) Adaptive Auxiliary Task Selection for Multitasking-Assisted Constrained Multi-Objective Optimization [Feature]. *IEEE Computational Intelligence Magazine*, **18**, 18-30. https://doi.org/10.1109/mci.2023.3245719

[11] Yuan, J., Liu, H., Ong, Y. and He, Z. (2022) Indicator-Based Evolutionary Algorithm for Solving Constrained Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, **26**, 379-391. https://doi.org/10.1109/tevc.2021.3089155

[12] Farias, L.R.C. and Araújo, A.F.R. (2024. An Inverse Modeling Constrained Multi-Objective Evolutionary Algorithm Based on Decomposition. 2024 *IEEE International Conference on Systems, Man, and Cybernetics* (*SMC*), Kuching, 6-10 October 2024, 3727-3732. https://doi.org/10.1109/smc54092.2024.10831275

[13] Tian, Y., Zhang, T., Xiao, J., Zhang, X. and Jin, Y. (2021) A Coevolutionary Framework for Constrained Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, **25**, 102-116. https://doi.org/10.1109/tevc.2020.3004012

[14] Ming, F., Gong, W., Wang, L. and Gao, L. (2023) A Constraint-Handling Technique for Decomposition-Based Constrained Many-Objective Evolutionary Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, **53**, 7783-7793. https://doi.org/10.1109/tsmc.2023.3299570

[15] Fan, Z., Li, W., Cai, X., Huang, H., Fang, Y., You, Y., *et al*. (2019) An Improved Epsilon Constraint-Handling Method in MOEA/D for CMOPs with Large Infeasible Regions. *Soft Computing*, **23**, 12491-12510. https://doi.org/10.1007/s00500-019-03794-x

[16] Zhou, Y., Xiang, Y. and He, X. (2021) Constrained Multiobjective Optimization: Test Problem Construction and Performance Evaluations. *IEEE Transactions on Evolutionary Computation*, **25**, 172-186. https://doi.org/10.1109/tevc.2020.3011829

[17] Bosman, P.A.N. and Thierens, D. (2003) The Balance between Proximity and Diversity in Multiobjective Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, **7**, 174-188. https://doi.org/10.1109/tevc.2003.810761

[18] Zitzler, E. and Thiele, L. (1999) Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, **3**, 257-271. https://doi.org/10.1109/4235.797969

附　录

Table A1. Wilcoxon rank-sum test results of IGD values obtained by all algorithms on 30 sets of test problems
表 A1. 所有算法在 30 组测试问题中获得的 IGD 值的 Wilcoxon 秩和检验统计结果

问题	C3M	CMOQLMT	IMCMOEAD	ICMA	CCMO	tDEACPBI	PFLCA
ZXH-CF1	7.93e−2−	6.70e−2−	9.36e−2−	5.30e−2−	4.76e−2−	4.89e−2−	**4.62e−2**
ZXH-CF2	2.09e−1−	5.06e−1−	3.96e−1−	3.95e−1−	1.64e−1=	2.70e−1−	**1.37e−1**
ZXH-CF3	1.39e−1−	1.49e−1−	1.26e−1−	1.11e−1−	7.58e−2−	1.01e−1−	**6.67e−2**
ZXH-CF4	1.53e+0−	1.16e+0−	1.24e+0−	1.42e+0−	2.51e−1=	5.02e−1−	**2.12e−1**
ZXH-CF5	1.22e−1−	4.08e−1−	2.25e−1−	2.72e−1−	1.94e−1−	1.55e−1−	**4.85e−2**
ZXH-CF6	4.80e−2−	4.30e−2−	5.62e−2−	3.70e−2−	3.35e−2−	4.54e−2−	**3.13e−2**
ZXH-CF7	5.53e−1−	6.89e−1−	7.46e−1−	7.23e−1−	1.74e−1−	2.47e−1−	**1.25e−1**
ZXH-CF8	1.12e−1−	2.19e−1−	1.64e−1−	2.24e−1−	6.05e−2−	8.80e−2−	**5.71e−2**
ZXH-CF9	2.19e−1−	1.10e−1−	2.97e−1−	1.72e−1−	**3.80e−2+**	1.32e−1−	5.05e−2
ZXH-CF10	1.45e+0−	1.35e+0−	1.35e+0−	1.94e+0−	**1.54e−1+**	6.01e−1−	1.83e−1
ZXH-CF11	7.66e−2−	4.65e−2−	8.42e−2−	4.89e−2−	**3.00e−2+**	6.10e−2−	3.78e−2
ZXH-CF12	2.71e−1−	6.71e−1−	7.71e−1−	6.45e−1−	1.56e−1=	5.21e−1−	**6.81e−2**
ZXH-CF13	6.73e−1−	1.04e+0−	1.13e+0−	1.28e+0−	3.11e−1=	**3.29e−1+**	4.00e−1
ZXH-CF14	4.27e−2−	4.84e−2−	5.01e−2−	1.02e−1−	1.64e−2−	2.64e−2−	**1.22e−2**
ZXH-CF15	**4.96e−2+**	1.77e−1−	3.10e−1−	2.70e−1−	8.22e−2−	1.76e−1−	5.42e−2
ZXH-CF16	5.51e−3−	4.78e−3−	2.19e−2−	5.94e−3−	3.21e−3=	2.42e−2−	**3.26e−3**
LIRCMOP1	**2.89e−1+**	3.08e−1+	3.57e−1=	3.07e−1+	3.07e−1+	3.15e−1+	3.54e−1
LIRCMOP2	2.77e−1+	2.72e−1+	3.02e−1+	**2.38e−1+**	2.75e−1+	2.55e−1+	3.41e−1
LIRCMOP3	3.61e−1=	3.18e−1=	3.42e−1=	3.12e−1=	3.18e−1=	**3.06e−1=**	3.39e−1
LIRCMOP4	3.58e−1=	2.89e−1=	3.16e−1=	2.84e−1=	2.97e−1=	**2.78e−1+**	2.92e−1
LIRCMOP5	2.52e+0−	1.55e+0−	2.64e+0−	2.41e+0−	7.86e−1−	1.61e+0−	**3.57e−1**
LIRCMOP6	2.64e+0−	1.42e+0−	2.84e+0−	2.33e+0−	1.07e+0−	1.46e+0−	**4.12e−1**
LIRCMOP7	2.34e+0−	1.42e+0−	3.33e+0−	1.77e+0−	1.73e−1−	1.61e+0−	**1.61e−1**
LIRCMOP8	2.38e+0−	1.54e+0−	3.44e+0−	2.01e+0−	3.46e−1−	1.68e+0−	**2.62e−1**
LIRCMOP9	**6.24e−1=**	8.74e−1−	1.45e+0−	7.50e−1−	9.37e−1−	1.06e+0−	6.41e−1
LIRCMOP10	8.20e−1−	9.30e−1−	1.21e+0−	8.51e−1−	9.81e−1−	1.06e+0−	**6.71e−1**
LIRCMOP11	6.13e−1=	8.49e−1−	1.33e+0−	7.09e−1−	7.20e−1−	9.96e−1−	**4.22e−1**
LIRCMOP12	5.39e−1−	7.57e−1−	1.22e+0−	6.58e−1−	6.51e−1−	9.85e−1−	**4.11e−1**
LIRCMOP13	1.36e+0−	1.33e+0−	1.39e+0−	1.32e+0−	2.29e−1−	1.31e+0−	**1.15e−1**
LIRCMOP14	1.31e+0−	1.29e+0−	1.35e+0−	1.28e+0−	1.87e−1−	1.27e+0−	**1.03e−1**
+/−/=	3/23/4	2/26/2	1/26/3	2/26/2	5/18/7	4/25/1	

Table A2. Wilcoxon rank-sum test results of HV values obtained by all algorithms on 30 sets of test problems
表 A2. 所有算法在 30 组测试问题中获得的 HV 值的 Wilcoxon 秩和检验统计结果

问题	C3M	CMOQLMT	IMCMOEAD	ICMA	CCMO	tDEACPBI	PFLCA
ZXH-CF1	7.84e−1−	8.02e−1−	7.35e−1−	8.20e−1−	8.28e−1−	8.26e−1−	**8.33e−1**
ZXH-CF2	3.04e−1−	1.31e−1−	1.41e−1−	1.73e−1−	4.43e−1−	3.19e−1−	**4.51e−1**
ZXH-CF3	3.68e−1−	3.52e−1−	4.01e−1−	4.11e−1−	4.74e−1−	4.58e−1−	**4.98e−1**
ZXH-CF4	5.54e−3−	5.46e−3−	1.24e−3−	9.87e−4−	2.05e−1−	5.50e−2−	**2.45e−1**
ZXH-CF5	1.87e−1−	6.34e−2−	1.55e−1−	1.03e−1−	1.92e−1−	1.98e−1−	**2.78e−1**
ZXH-CF6	1.97e−1−	2.04e−1−	2.06e−1−	2.13e−1−	2.18e−1−	2.07e−1−	**2.23e−1**
ZXH-CF7	3.60e−3−	1.78e−5−	2.22e−3−	8.85e−4−	1.68e−1=	1.22e−1−	**2.24e−1**
ZXH-CF8	1.46e−1−	8.07e−2−	1.02e−1−	8.12e−2−	**1.95e−1+**	1.65e−1−	1.97e−1
ZXH-CF9	8.23e−2−	1.21e−1−	1.15e−1−	8.78e−2−	**2.30e−1+**	1.78e−1−	2.06e−1
ZXH-CF10	2.47e−4−	4.29e−4−	1.56e−3−	0.00e+0−	**1.68e−1+**	6.01e−2−	1.29e−1
ZXH-CF11	2.95e−1−	3.72e−1−	3.39e−1−	3.48e−1−	**4.13e−1+**	3.69e−1−	4.01e−1
ZXH-CF12	3.71e−1−	1.18e−1−	1.48e−1−	1.87e−1−	5.67e−1−	2.53e−1−	**6.56e−1**
ZXH-CF13	1.23e−2−	3.43e−3−	9.70e−4−	1.22e−3−	7.51e−2=	**1.16e−1+**	1.32e−1
ZXH-CF14	4.69e−1−	4.60e−1−	4.54e−1−	4.41e−1−	5.17e−1=	4.99e−1−	**5.22e−1**
ZXH-CF15	5.90e−1−	4.24e−1−	3.86e−1−	4.01e−1−	5.64e−1=	4.87e−1−	**5.99e−1**
ZXH-CF16	7.68e−1−	7.70e−1−	7.70e−1−	7.67e−1−	7.75e−1=	7.60e−1−	**7.75e−1**
LIRCMOP1	**1.54e−1+**	1.43e−1+	9.95e−2=	1.11e−1=	1.10e−1=	1.08e−1=	1.01e−1
LIRCMOP2	2.21e−1=	2.21e−1=	2.14e−1=	**2.35e−1+**	2.20e−1=	2.29e−1+	2.13e−1
LIRCMOP3	8.42e−2=	9.76e−2=	9.28e−2=	1.00e−1=	9.67e−2=	**1.00e−1+**	9.59e−2
LIRCMOP4	1.72e−1−	1.92e−1=	1.84e−1=	1.99e−1=	1.91e−1=	**1.94e−1+**	2.03e−1
LIRCMOP5	3.74e−3−	0.00e+0−	0.00e+0−	0.00e+0−	6.81e−2−	0.00e+0−	**1.36e−1**
LIRCMOP6	3.50e−3−	9.86e−4−	0.00e+0−	0.00e+0−	2.43e−2−	5.99e−2−	**9.03e−2**
LIRCMOP7	1.48e−2−	3.81e−2−	0.00e+0−	0.00e+0−	2.32e−1=	8.63e−3−	**2.32e−1**
LIRCMOP8	1.34e−2−	2.09e−2−	0.00e+0−	0.00e+0−	2.01e−1=	0.00e+0−	**2.14e−1**
LIRCMOP9	**2.74e−1+**	1.44e−1−	1.40e−2−	2.23e−1=	1.44e−1−	8.40e−2−	2.74e−1
LIRCMOP10	1.91e−1=	1.05e−1−	1.67e−2−	1.28e−1−	6.31e−2−	4.93e−2−	**2.25e−1**
LIRCMOP11	3.39e−1−	1.81e−1−	4.61e−2−	2.55e−1−	2.60e−1−	1.57e−1−	**3.97e−1**
LIRCMOP12	3.36e−1−	2.22e−1−	1.11e−1−	2.81e−1−	3.35e−1−	1.91e−1−	**4.07e−1**
LIRCMOP13	4.59e−5−	1.05e−4−	7.79e−6−	1.10e−4−	3.98e−1−	2.98e−4−	**5.23e−1**
LIRCMOP14	2.19e−3−	3.04e−3−	6.56e−5−	4.94e−4−	4.59e−1−	8.12e−4−	**5.53e−1**
+/−/=	2/25/3	1/26/3	0/26/4	1/25/4	4/15/11	4/25/1	

Table A3. Wilcoxon rank-sum test results of FR values obtained by all algorithms on 30 sets of test problems
表 A3. 所有算法在 30 组测试问题中获得的 FR 值的 Wilcoxon 秩和检验统计结果

问题	C3M	CMOQLMT	IMCMOEAD	ICMA	CCMO	tDEACPBI	PFLCA
ZXH-CF1	1.000=	1.000=	1.000=	1.000=	1.000=	1.000=	**1.000**
ZXH-CF2	1.000=	1.000=	0.989=	0.967−	**1.000=**	0.966−	0.999
ZXH-CF3	1.000=	1.000=	1.000=	1.000=	1.000=	1.000=	**1.000**
ZXH-CF4	0.000−	1.000+	0.673−	0.887=	**1.000+**	1.000+	0.902
ZXH-CF5	**1.000=**	1.000=	0.995=	0.867−	1.000=	1.000=	1.000
ZXH-CF6	1.000=	1.000=	0.997=	1.000=	1.000=	1.000=	**1.000**
ZXH-CF7	0.002−	0.987=	0.497−	0.615−	1.000=	1.000=	**1.000**
ZXH-CF8	1.000=	1.000=	0.764−	**1.000=**	0.985−	0.943−	1.000
ZXH-CF9	**1.000=**	1.000=	0.527−	1.000=	1.000=	1.000=	1.000
ZXH-CF10	0.000−	0.987=	0.147−	0.541−	1.000=	1.000=	**1.000**
ZXH-CF11	1.000=	1.000=	0.991=	1.000=	1.000=	1.000=	**1.000**
ZXH-CF12	1.000=	0.967−	0.975=	0.967=	1.000=	**1.000=**	9.983
ZXH-CF13	0.212−	0.967−	0.467−	0.756−	1.000=	1.000=	**1.000**
ZXH-CF14	1.000=	0.967=	0.819−	0.970=	**1.000=**	1.000=	9.816
ZXH-CF15	1.000=	0.967−	0.995=	0.937−	0.967−	1.000=	**1.000**
ZXH-CF16	1.000=	1.000=	0.840−	1.000=	1.000=	**1.000**=	1.000
LIRCMOP1	0.173−	0.956=	0.324−	1.000+	**1.000+**	0.992+	0.951
LIRCMOP2	0.021−	0.970=	0.791−	1.000+	**1.000+**	0.981+	0.894
LIRCMOP3	0.005−	0.969=	0.714−	**1.000+**	1.000+	1.000+	0.918
LIRCMOP4	0.004−	0.992=	0.781−	1.000+	**1.000+**	0.984=	0.976
LIRCMOP5	1.000=	1.000=	1.000=	1.000=	1.000=	1.000=	1.000
LIRCMOP6	1.000=	1.000=	1.000=	1.000=	1.000=	1.000=	**1.000**
LIRCMOP7	1.000=	1.000=	1.000=	1.000=	1.000=	1.000=	**1.000**
LIRCMOP8	1.000=	1.000=	1.000=	1.000=	**1.000=**	1.000=	1.000
LIRCMOP9	1.000=	1.000=	1.000=	1.000=	**1.000=**	1.000=	1.000
LIRCMOP10	1.000=	1.000=	1.000=	1.000=	**1.000=**	1.000=	1.000
LIRCMOP11	1.000=	1.000=	1.000=	1.000=	**1.000=**	1.000=	1.000
LIRCMOP12	1.000=	1.000=	1.000=	1.000=	1.000=	1.000=	**1.000**
LIRCMOP13	1.000=	1.000=	1.000=	1.000=	1.000=	1.000=	**1.000**
LIRCMOP14	1.000=	1.000=	1.000=	1.000=	**1.000=**	1.000=	1.000
+/−/=	0/8/22	1/3/26	0/12/18	4/6/20	5/2/23	4/2/24	