Advances in Applied Mathematics 应用数学进展, 2026, 15(3), 95-109
Published Online March 2026 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2026.153091

Hans汉斯

基于多归档协同进化的约束多目标优化算法研究

张凤韬，陈　磊*

广东工业大学数学与统计学院，广东 广州

收稿日期：2026年2月4日；录用日期：2026年2月27日；发布日期：2026年3月5日

摘　要

近些年以来，约束多目标优化问题在真实的能源系统以及工程问题中广泛应用，但是这类问题在求解过程中可能遇到可行域狭窄不连续，收敛速度慢，解集分布不均衡等问题使得求解这类问题变得困难。为解决求解中所遇到的问题，本文提出一种增强型多归档协同进化算法Enhanced Multi-Archive Cooperative Evolutionary Algorithm (EMFEACD)，该算法利用收敛、多样性以及可行性归档，并引入技能因子进行跨归档知识迁移。最后通过自适应调整算子动态平衡搜索的探索与开发强度。并在MW与LIR-CMOP总计28个典型测试问题中的大多数问题测试结果上优于现有主流算法。

关键词

约束多目标优化，协同进化，差分进化，归档策略，遗传算法

Research on Constrained Multi-Objective Optimization Algorithm Based on Multi-Archiving Collaborative Evolution

Fengtao Zhang, Lei Chen*

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

Received: February 4, 2026; accepted: February 27, 2026; published: March 5, 2026

Abstract

In recent years, constrained multi-objective optimization problems have been widely applied in

*通讯作者。

文章引用: 张凤韬, 陈磊. 基于多归档协同进化的约束多目标优化算法研究[J]. 应用数学进展, 2026, 15(3): 95-109.
DOI: 10.12677/aam.2026.153091

real-world energy systems and engineering challenges. However, solving these problems often encounters difficulties such as narrow and discontinuous feasible regions, slow convergence rates, and uneven solution distribution. To address these issues, this paper proposes an enhanced multi-archive cooperative evolutionary algorithm (EMFEACD). The algorithm utilizes convergence, diversity, and feasibility archiving, while introducing a skill factor to facilitate knowledge transfer across archives. Additionally, it dynamically adjusts the operator to balance exploration and development intensity. The proposed algorithm outperforms existing mainstream algorithms in most of the 28 typical test problems, including MW and LIR-CMOP.

Keywords

Constrained Multi-Objective Optimization, Co-Evolution, Differential Evolution, Archiving Strategy, Genetic Algorithm

1. 引言

多目标优化问题广泛应用于工程设计、能源的调度、智能领域的制造中，而在现实问题中往往有非常多的约束条件，约束多目标优化问题(Constrained Multi-Objective Optimization Problems, CMOPs)由此出现[1] [2]。与没有约束的情况相比，CMOPs 的求解不仅要权衡各个目标之间的冲突，还要应对可行域的不规则以及不连续的复杂结构。在搜索中如果无法准确的找到可行域或无法稳定的逼近 Pareto 前沿，优化效果会显著下降，所以如何在收敛性、多样性、和可行性三者中建立有效的权衡是研究 CMOPs 问题的核心挑战。

受到各种复杂约束结构的影响，传统的优化方法在高维、可行域分割不连续的情况下性能往往不理想。近年来，进化算法由于无需计算梯度，并行优化以及适应能力强等优势被广泛用来求解 CMOPs 问题，典型的方法包括基于约束支配原则(Constraint Domination Principle, CDP)的 NSGA-II-CDP 算法、MOEA/D-CDP 算法[3]以及结合参考点的 C-NSGA-III 算法等，这些方法成功的减弱了惩罚因子的敏感性，同时提高了可行性维持能力。然而当可行域特别狭窄或目标冲突显著时，仅仅倚靠单一的约束处理策略依旧难以保证收敛速度和解多样性。

为了进一步提升解决 CMOPs 问题算法的性能，有研究引入了多归档结构、多任务学习以及协同进化等方法。例如，多归档策略通过不仅关注解的收敛性能的同时还关注解的多样性，从而优化了解的分布[4] [5]。PPS(Push and Pull Search)算法通过推和拉分别让解大胆探索和向前沿逼近[6]；协同进化和多任务进化方法则通过辅助种群或技能因子实现跨个体的知识迁移策略[7] [8]。虽然这些方法提升了算法可行性处理的能力或多样性维持的水平，但是仍然存在以下不足：第一，多数的归档方法只是采用了并列式的归档结构，而归档之间缺乏信息的交流合作；第二，归档之间优质解的迁移效率低下，知识利用率较差；第三，搜索算子在不同阶段缺少自适应调节，难以在全局开发之间实现搜索的动态平衡；第四，还有部分方法对参数过度依赖，减低了算法的泛化性。针对以上问题本文提出一种增强型多归档协同进化算法(Enhanced Multi-Archive Cooperative Evolutionary Algorithm with Differential/GA Operators, EMFEACD)，该算法旨在同时提升收敛性，可行性和解集多样性。算法构建了前向探索、多样性增强、与可行性利用三类功能性归档，形成具有明确分工且具有协作机制的多归档结构，并利用归档性能反馈的调度策略在

不同的搜索阶段自适应的选择合适的算子，从而实现探索与开发的动态平衡；最后引入技能因子及权重调节机制，使优质解能够在归档之间选择性的迁移，以提升可行域探索效率并抑制早熟收敛。

本文的主要贡献如下：

(1) 提出一种多归档协同机制，将收敛、多样性与可行性分为不同归档，并通过性能反馈实现归档间的知识迁移，提高复杂约束条件下的求解效率。

(2) 设计差分进化与遗传算子的混合框架，并基于归档表现实现算子的自适应调度，使搜索过程能够动态平衡全局探索与局部收敛。

(3) 引入技能因子与权重调节机制，实现跨归档的学习偏好引导，促进优质解共享，加快可行域探索并提升解集质量。

通过在 MW 与 LIR-CMOP 等典型基准问题上的实验，验证了 EMFEACD 算法在收敛速度、可行性维持与 Pareto 前沿覆盖能力方面的优势，并展示了其在复杂约束场景下的稳定性与鲁棒性。

2. 相关工作

对约束多目标优化算法的研究主要集中在约束的处理方法、进化框架设计以及归档的设计和管理制的方向，虽然现在已有的方法已经取得了不错的进展，但在处理可行域高度复杂、约束耦合紧密或 P 前沿形状比较复杂时仍然会出现收敛速度慢，可行性和多样性不足的问题。

传统罚函数方法通过在目标函数中嵌入惩罚项实现可行性引导，但罚因子的取值高度依赖人工经验且对不同问题场景具有强敏感性，极易导致可行解难以生成或搜索过程陷入局部停滞。为改善这一缺陷，诸多研究引入约束支配原则(Constraint Domination Principle, CDP)，典型代表如 NSGA-II-CDP 与 MOEA/D-CDP，该类方法通过在支配关系定义中显式区分可行解与不可行解的优先级，有效减弱了对罚因子的依赖敏感性。然而，当可行域极度狭窄或不可行区域呈现多峰分布结构时，仅依靠 CDP 单一约束处理策略，仍难以保证算法在复杂搜索空间中维持稳定的可行性探索能力。

在进化框架与协同机制方面，为提升复杂约束场景下的搜索效率，研究者逐步引入协同进化与多任务学习的核心思想。代表性方法如 CCMO 算法通过构建主群体与辅助群体的协作模式，助力算法跨越大范围不可行区域；而多任务进化框架(MFEA)则借助技能因子(Skill Factor)实现跨任务的知识共享与迁移[9] [10]，显著提升了搜索过程的效率。但此类方法存在明显局限：当不同任务或子种群之间的差异较大时，知识迁移极易失效甚至产生负迁移效应，反而阻碍搜索进程；同时，该类方法对种群的初始结构与协作模式依赖度较高，难以在不同特征的约束多目标优化问题间保持稳健的优化表现。

归档管理作为多目标优化中维持解集收敛性与多样性的核心技术，受到广泛关注。Two-Archive 系列方法通过分别构建收敛归档与多样性归档，并采用并行更新机制[11]，使算法在 Pareto 前沿逼近与解集覆盖性之间建立基本平衡；PPS (Push and Pull Search)方法则通过“推–拉”两阶段搜索机制，分别强化全局探索能力与可行性处理效率，在部分复杂结构的 CMOPs 中取得了较好的优化效果。然而，现有多归档策略仍存在显著不足：一方面，不同归档多以“并列独立”的方式运行，缺乏精细化的协作机制，优质解在各归档之间的迁移效率低下，难以实现动态互补；另一方面，搜索算子在不同优化阶段的适配性调整不足，导致全局探索与局部收敛之间难以建立自适应的平衡关系，影响算法的整体优化性能。

综上，现有约束多目标优化算法在处理复杂 CMOPs 时仍面临三大核心挑战：① 多归档之间缺乏高效协作机制，导致优质解的利用效率不足；② 可行性、收敛性与多样性三者之间的动态平衡机制欠缺，难以适配复杂多变的约束结构；③ 搜索算子的阶段性适应调整不够灵活，无法根据优化进程动态匹配探索与开发需求。为有效解决这些问题，本文提出增强型多归档协同进化算法(Enhanced Multi-Archive

Cooperative Evolutionary Algorithm, EMFEACD)，通过构建多归档协同机制、设计基于性能反馈的算子自适应调度策略以及引入技能因子实现跨归档知识迁移，旨在全面提升算法在复杂约束环境下的收敛效率、可行性保障能力及解集多样性。

3. 算法设计

本节将详细阐述增强型多归档协同进化算法 EMFEACD 算法。该算法的核心思想在于构建分工明确的多归档结构，根据性能反馈实现算子的自适应调度与跨归档的知识迁移，进而使得算法在收敛性、可行性与多样性之间形成动态平衡。

3.1. 算法总体框架

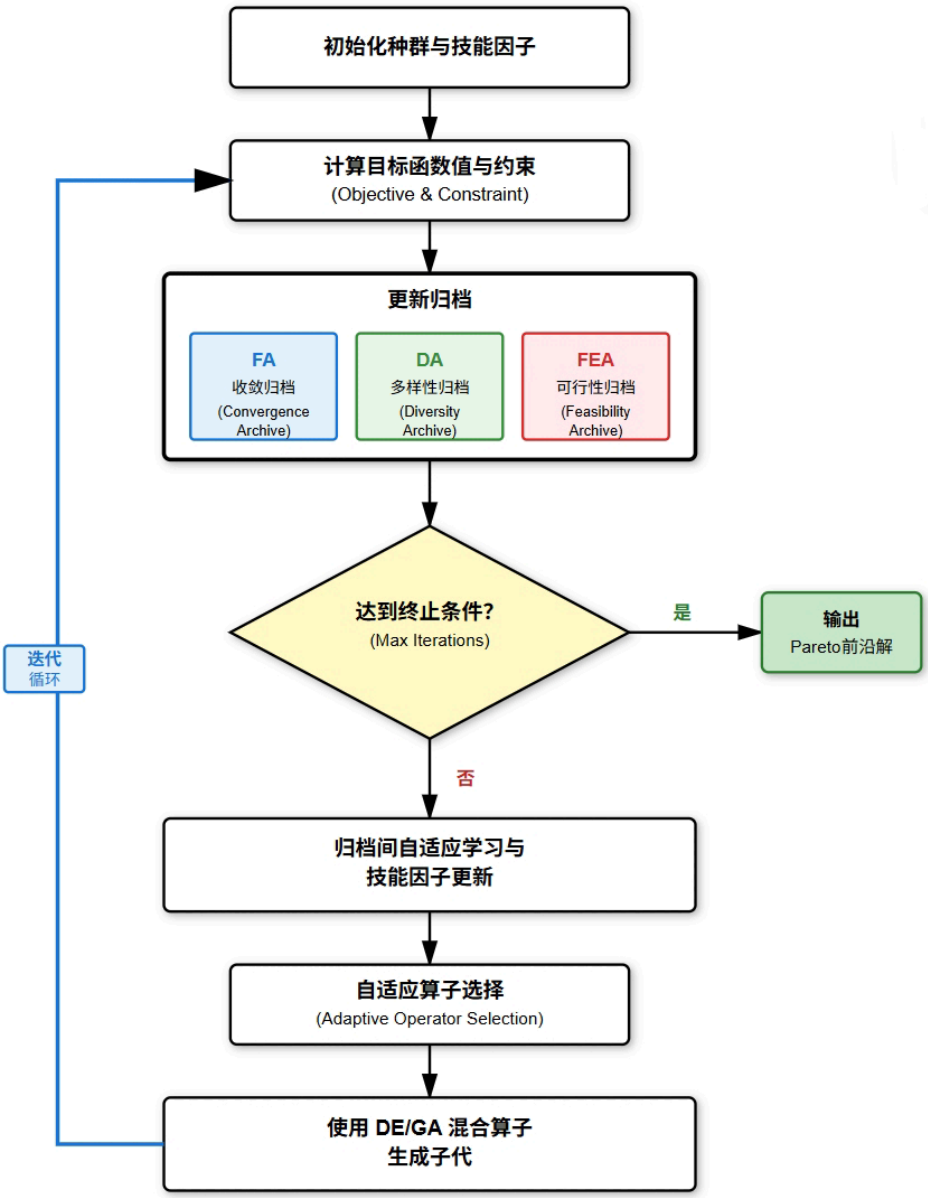

Figure 1. Flowchart of EMFEACD algorithm
图 1. EMFEACD 算法流程框架

EMFEACD 算法同时维护三种功能化归档：前向探索归档(Forward Exploration Archive, FA)、多样性增强归档(Diversity Archive, DA)与可行性利用归档(Feasibility Exploitation Archive, FEA) [12]，分别承担着收敛推进、多样性保持和可行性提升的职责，并在进化过程中进行演化。算法整体流程如图 1 所示，个体根据技能因子从匹配度更高的档案选取父代，并在差分进化(DE)与遗传算法(GA)算子间自适应选择生成子代[13]；随后按照可行性、支配关系和多样性准则更新各个归档；最后通过权重驱动的归档协作机制实现优质解迁移。

下面是提出算法的伪代码。

算法 1 所提 EMFEACD 算法总框架

输入：种群规模 N，最大评价次数 MaxFEs
输出：近似 Pareto 最优解集
1: 初始化种群 P，构建 FA、DA、FEA
2: 为每个个体分配初始技能因子 sf(i)
3: while FEs < MaxFEs do
4: 依据归档性能自适应更新算子权重 wpDE,wpGA
5: 对每个归档并行执行：
6: 依据技能因子选择父代
7: 按算子权重选择 DE 或 GA 生成子代
8: 根据约束与支配关系更新归档
9: 依据归档性能差异触发归档间迁移
10: 更新技能因子与归档权重
11: end while
12: 返回合并后的非支配解集

3.2. 多归档策略设计

(1) 前向探索归档 FA：提升收敛速度

FA 的核心任务是引导搜索向 Pareto 前沿逼近。对于解 x 其目标向量 $F(x)=(f_1(x),\cdots,f_m(x))$ 到动态理想点 z^* 的距离定义为：

$$d(x)=\left\|F(x)-z^*\right\|_2 \tag{1}$$

$$z_j^*=\min_{x\in P} f_j(x) \tag{2}$$

FA 更新时优先保留非支配解，并按 $d(x)$ 升序排序，从而建立稳定的收敛梯度。

(2) 多样性增强归档 DA：保持解集覆盖

DA 采用最大最小距离 Max-Min 策略提升目标空间中的分布均匀性。

设已选解集为 S，候选解为 x 的多样性度量为：

$$D(x)=\min_{y\in S}\left\|F(x)-F(y)\right\|_2 \tag{3}$$

DA 每次遴选使 $D(x)$ 最大的解进入归档，防止解集过度聚集。

(3) 可行性利用归档 FEA：提升可行解比例

个体 x 的约束违反度(Constraint Violation, CV)定义为：

$$CV(x)=\sum_{i=1}^{q}\max(0,g_i(x))+\sum_{j=1}^{p}\left|h_j(x)\right| \tag{4}$$

FEA 更新规则采用改进 CDP：

- 可行解始终优于不可行解；
- 不可行解按 CV 值升序保留；
- 可行解内部采用非支配排序与拥挤距离筛选。

FEA 保证算法不偏离可行域太远，并在后期提供收敛引导。

3.3. 混合操作算子与自适应调度

(1) 差分进化算子(DE)：增强全局探索

DE 操作通过差分向量生成变异个体：

$$v = a + F \cdot (b - c) \tag{5}$$

随后通过二项式交叉生成子代：

$$u_j = \begin{cases} v_j, & \text{if } rand_j < CR \\ x_j, & \text{otherwise} \end{cases} \tag{6}$$

DE 擅长跳出局部区域，适合探索可行域边界与复杂结构。

(2) 遗传算子(GA)：增强局部开发

GA 采用均匀交叉与小幅变异：

- 交叉概率 $(p_c \approx 0.9)$；
- 变异概率 $(p_m = 1/D)$ (D 为维数)。

该策略在中后期可提升解的精细收敛能力。

(3) 算子自适应调度机制

记最近 T 代中归档性能提升量为(ΔFA, ΔDA, ΔFEA)，算子 DE 与 GA 在各归档的贡献为 $(s_{\mathrm{DE}}, s_{\mathrm{GA}})$。算子权重更新方式如下：

$$wp_k(t+1) = (1-\eta) \cdot wp_k(t) + \eta \cdot \frac{s_k}{s_{\mathrm{DE}} + s_{\mathrm{GA}}}, k \in \{\mathrm{DE}, \mathrm{GA}\} \tag{7}$$

其中 η 为学习率。归一化后的权重决定每次子代生成时的算子选择概率，使算法自动适配不同搜索阶段。

3.4. 技能因子与权重自适应学习

每个个体 x 拥有技能因子 $sf(x) \in \mathrm{FA}, \mathrm{DA}, \mathrm{FEA}$，标识其最适配的学习归档。

(1) 归档性能度量

三类归档采用不同评估指标：

- P_{FA}：收敛距离下降幅度；
- P_{DA}：多样性指标变化量；
- P_{FEA}：可行解比例变化或平均 CV 缩减量。

性能经 Min-Max 归一化：

$$\tilde{P}_i = \frac{P_i - \min(P)}{\max(P) - \min(P)} \tag{8}$$

$$w_i(t+1) = (1-\alpha) w_i(t) + \alpha \tilde{P}_i \tag{9}$$

权重用于引导技能因子分配与迁移决策。

(2) 技能因子的功能

技能因子影响三方面：

- 父代选择时倾向从对应归档抽取；
- 跨归档迁移时的候选个体筛选；
- 归档表现越优，越多个体向其学习，形成正向反馈循环。

3.5. 归档间协同机制

为提升跨归档知识利用效率，EMFEACD 每隔固定代数执行协同迁移：

(1) 源归档与目标归档确定

- 源归档取当前性能最优者；
- 目标归档取性能最弱者。

(2) 迁移个体选择

- 从 FA 选取收敛最强个体(距理想点最近)；
- 从 DA 选取多样性最强个体(间距最大)；
- 从 FEA 选取可行度最高个体(CV 最小)。

(3) 目标归档更新

迁移解与目标归档合并后，经非支配排序与拥挤距离排序保留，确保归档质量不降级。该协同机制实现了跨模块的最优解复用，有助于突破可行域边界、改善分布并加速收敛。

3.6. 复杂度分析

设种群规模为 N，归档规模为 M，目标数为 m。各操作复杂度如下：

- 归档更新：$\left(O\left(M^2m\right)\right)$ (非支配排序)；
- 子代生成(DE/GA)：$\left(O\left(ND\right)\right)$；
- 性能评估与权重更新：$\left(O\left(N\right)\right)$；
- 归档协作迁移：$\left(O\left(M\log M\right)\right)$。

总体复杂度约为$\left(O\left(ND+M2m\right)\right)$，主要开销来自三归档的独立更新操作。

4. 实验与结果

4.1. 统计显著性检验

为全面评估 EMFEACD 算法性能，选用 MW 与 LIR-CMOP 共 28 个基准函数，与 NSGA-II、CMOEAD、PPS、CTAEA [14]、NSGA-III [15]进行对比，实验设置种群规模 100、最大评价次数 100,000、三类归档容量均为 100、$\eta=0.1$、$\alpha=0.2$、迁移间隔 10 代，分别从 IGD、HV、可行性率和运行时间四方面评估，结果见表 1~4。IGD 指标(表 1)显示 EMFEACD 在 24/28 个问题上取得最优，在 MW1/2/5/6/9/10/13/14 及 LIRCMOP3/4/6/7/8/9 等复杂约束问题上收敛精度显著领先，仅 MW4/7/8 等少数问题表现相当且标准差普遍较小，稳定性强；HV 指标(表 2)显示绝大多数问题上获最优 HV、前沿覆盖与多样性保持良好，仅 LIRCMOP7/8 略逊；可行性率(表 3)显示全部 28 个问题均保持 100%可行率，显著优于 CTAEA(部分问题不足 5%)，FEA 归档有效维持可行性；运行时间(表 4)显示为 1.3~10.2 秒，高于 NSGA-II 但显著低于 CTAEA(30-69 秒)，多归档机制效率可接受。综上，EMFEACD 在收敛速度、前沿覆盖与可行性维持方面优势突出，验证了多归档协同与自适应调度机制的有效性。

Table 1. Comparison of IGD metrics of various algorithms on test problems
表 1. 各算法在测试问题上的 IGD 指标比较

Problem		CMOEAD	NSGA-II	PPS	CTAEA	NSGA-III	EMFEACD
MW1	mean(std)	1.0883e−2 (1.55e−2) −	2.5574e−3 (1.81e−3) −	1.2304e−1 (1.84e−1) −	2.0136e−3 (5.50e−5) −	1.0117e−2 (1.37e−2) −	**1.7770e−3 (7.16e−5)**
MW2	mean(std)	2.3458e−2 (8.46e−3) −	2.9518e−2 (1.09e−2) −	1.7666e−1 (2.01e−1) −	1.7534e−2 (6.49e−3) −	2.5166e−2 (7.73e−3) −	**4.9205e−3 (4.56e−4)**
MW3	mean(std)	6.5697e−3 (7.71e−4) −	5.8711e−3 (3.71e−4) −	6.4126e−3 (4.01e−4) −	5.5124e−3 (5.76e−4) −	5.5406e−3 (4.23e−4) −	**5.1383e−3 (1.89e−4)**
MW4	mean(std)	4.2860e−2 (2.48e−3) +	5.7998e−2 (2.41e−3) −	6.4918e−2 (9.22e−3) −	4.6538e−2 (4.08e−4) =	**4.1484e−2 (1.66e−4) +**	4.6739e−2 (9.62e−4)
MW5	mean(std)	4.9443e−3 (3.41e−3) −	3.0511e−1 (3.78e−1) −	1.7597e−1 (2.96e−1) −	1.6086e−2 (3.76e−3) −	3.5182e−1 (3.55e−1) −	**2.0550e−3 (5.26e−4)**
MW6	mean(std)	5.8189e−2 (1.31e−1) −	2.3254e−2 (7.65e−3) −	4.4664e−1 (3.16e−1) −	8.0931e−3 (6.56e−3) −	1.0426e−1 (1.73e−1) −	**2.7947e−3 (5.83e−5)**
MW7	mean(std)	**5.0258e−3 (2.90e−4) +**	9.4375e−2 (1.88e−1) =	5.6739e−3 (3.63e−4) =	7.5816e−3 (9.55e−4) −	5.7344e−3 (4.09e−4) =	5.4652e−3 (3.40e−4)
MW8	mean(std)	5.2411e−2 (2.65e−3) =	6.5271e−2 (7.69e−3) −	1.3127e−1 (6.18e−2) −	5.7116e−2 (6.46e−3) −	5.4034e−2 (7.02e−3) =	**5.0921e−2 (2.83e−3)**
MW9	mean(std)	1.1596e−2 (2.99e−3) −	1.0851e−2 (4.52e−3) −	3.4561e−1 (3.04e−1) −	9.0193e−3 (7.26e−4) −	1.8900e−1 (2.81e−1) −	**5.3197e−3 (4.34e−4)**
MW10	mean(std)	5.9836e−2 (5.47e−2) −	9.9866e−2 (7.90e−2) −	3.3153e−1 (2.24e−1) −	1.7346e−2 (1.03e−2) −	1.7158e−1 (1.76e−1) −	**3.5415e−3 (9.95e−5)**
MW11	mean(std)	3.5860e−1 (3.24e−1) −	2.5662e−1 (3.30e−1) =	**7.2233e−3 (2.22e−4) +**	1.6552e−2 (1.71e−3) −	6.6479e−1 (1.05e−1) −	7.5403e−3 (2.83e−4)
MW12	mean(std)	**5.2729e−3 (6.33e−4) =**	5.6076e−3 (3.34e−4) −	2.9317e−1 (3.07e−1) −	8.1449e−3 (5.00e−4) −	2.6803e−1 (3.36e−1) −	5.2775e−3 (1.96e−4)
MW13	mean(std)	1.0329e−1 (4.32e−2) −	1.0656e−1 (2.13e−2) −	5.3192e−1 (4.01e−1) −	4.2566e−2 (2.62e−2) −	3.5272e−1 (4.27e−1) −	**1.6260e−2 (6.00e−3)**
MW14	mean(std)	2.1158e−1 (2.96e−3) −	1.2721e−1 (3.15e−3) −	1.6557e−1 (4.41e−2) −	1.2673e−1 (4.28e−2) −	1.4226e−1 (1.99e−2) −	**1.0239e−1 (2.29e−3)**
LIRCMOP1	mean(std)	2.7102e−1 (2.90e−2) −	3.0452e−1 (3.71e−2) −	7.6575e−2 (5.56e−2) =	2.6971e−1 (1.03e−1) −	3.1970e−1 (2.98e−2) −	**4.1979e−2 (1.14e−2)**
LIRCMOP2	mean(std)	2.4968e−1 (3.78e−2) −	2.6005e−1 (1.19e−2) −	**2.7178e−2 (1.77e−2) +**	2.1923e−1 (1.05e−1) −	2.6417e−1 (3.08e−2) −	3.7234e−2 (7.02e−3)
LIRCMOP3	mean(std)	2.7664e−1 (5.21e−2) −	3.2148e−1 (2.62e−2) −	1.1392e−1 (6.82e−2) −	3.1490e−1 (1.01e−1) −	3.0126e−1 (3.94e−2) −	**3.7813e−2 (1.39e−2)**
LIRCMOP4	mean(std)	2.8023e−1 (3.07e−2) −	2.8679e−1 (3.21e−2) −	5.1002e−2 (5.32e−2) =	2.8411e−1 (6.51e−2) −	2.9070e−1 (2.56e−2) −	**3.6711e−2 (2.16e−2)**
LIRCMOP5	mean(std)	1.3606e+0 (4.27e−1) −	1.2182e+0 (1.06e−2) −	**1.4032e−2 (1.67e−2) +**	1.2294e+0 (8.10e−3) −	1.2297e+0 (5.06e−3) −	2.9864e−2 (1.21e−2)
LIRCMOP6	mean(std)	1.3464e+0 (5.41e−4) −	1.3458e+0 (2.11e−4) −	7.7146e−2 (6.23e−2) =	1.3466e+0 (1.22e−3) −	1.3458e+0 (2.09e−4) −	**1.7804e−2 (3.54e−3)**
LIRCMOP7	mean(std)	1.5398e+0 (4.55e−1) −	9.1449e−1 (8.10e−1) −	1.2253e−1 (1.94e−2) =	5.1994e−1 (6.29e−1) −	7.8154e−1 (7.76e−1) −	**6.8175e−2 (6.21e−2)**
LIRCMOP8	mean(std)	1.6830e+0 (1.19e−3) −	9.6860e−1 (7.53e−1) −	1.9898e−1 (1.70e−2) −	7.8190e−1 (6.61e−1) −	1.4423e+0 (5.12e−1) −	**8.1882e−2 (6.96e−2)**

续表

LIRCMOP9	mean(std)	8.9715e−1 (1.38e−1) −	1.0038e+0 (9.51e−2) −	4.0144e−1 (7.82e−2) =	6.0242e−1 (6.56e−2) −	1.0391e+0 (2.75e−2) −	**3.1587e−1 (1.17e−1)**
LIRCMOP10	mean(std)	8.0763e−1 (1.53e−1) −	8.9706e−1 (5.88e−2) −	3.1489e−1 (1.42e−1) =	4.9328e−1 (1.12e−1) −	9.1391e−1 (1.14e−1) −	**1.8839e−1 (6.40e−2)**
LIRCMOP11	mean(std)	9.0345e−1 (8.38e−2) −	7.6286e−1 (1.29e−1) −	2.7375e−1 (2.54e−1) =	2.7090e−1 (1.32e−1) −	8.7625e−1 (1.47e−1) −	**5.2608e−2 (5.44e−2)**
LIRCMOP12	mean(std)	7.1400e−1 (1.68e−1) −	8.6606e−1 (1.65e−1) −	2.2542e−1 (7.86e−2) −	3.2906e−1 (1.42e−1) −	9.3560e−1 (8.76e−2) −	**1.1399e−1 (4.88e−2)**
LIRCMOP13	mean(std)	1.3048e+0 (4.47e−4) −	1.3261e+0 (3.44e−3) −	1.5106e−1 (2.07e−2) =	**1.1015e−1 (3.35e−3) +**	1.3052e+0 (5.58e−4) −	1.9286e−1 (4.63e−2)
LIRCMOP14	mean(std)	1.2611e+0 (4.00e−4) −	1.2821e+0 (2.40e−3) −	1.2360e−1 (6.86e−3) +	**1.1190e−1 (3.36e−3) +**	1.2617e+0 (6.78e−4) −	1.6961e−1 (3.86e−2)

注释 1. 算法在测试问题上的 IGD 指标平均值(+ 表示优于本算法，− 表示劣于本算法)。

Table 2. Comparison of HV metrics of various algorithms on test problems
表 2. 各算法在测试问题上的 HV 指标比较

Problem		CMOEAD	NSGA-II	PPS	CTAEA	NSGA-III	EMFEACD
MW1	mean(std)	1.2094e−1 (8.82e−3) −	1.1113e−1 (9.71e−3) −	2.0348e−1 (2.95e−2) =	1.2163e−1 (3.08e−2) −	1.0626e−1 (9.72e−3) −	**2.2166e−1 (5.49e−3)**
MW2	mean(std)	2.2875e−1 (1.75e−2) −	2.2958e−1 (8.51e−3) −	3.4879e−1 (1.12e−2) +	2.6436e−1 (2.96e−2) −	2.2460e−1 (1.33e−2) −	**3.4137e−1 (4.65e−3)**
MW3	mean(std)	1.0765e−1 (1.40e−2) −	9.8673e−2 (8.04e−3) −	1.6098e−1 (2.53e−2) −	1.0641e−1 (1.39e−2) −	1.0217e−1 (1.18e−2) −	1.9192e−1 (5.84e−3)
MW4	mean(std)	1.9614e−1 (1.42e−2) −	1.9703e−1 (1.26e−2) −	2.9432e−1 (2.10e−2) =	1.9650e−1 (2.49e−2) −	1.8928e−1 (1.09e−2) −	2.9839e−1 (9.93e−3)
MW5	mean(std)	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	2.8566e−1 (1.28e−2) +	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	**2.8029e−1 (4.02e−3)**
MW6	mean(std)	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	1.7761e−1 (1.60e−2) =	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	**1.9072e−1 (1.82e−3)**
MW7	mean(std)	2.1462e−2 (6.79e−2) −	1.2028e−1 (1.27e−1) −	2.4604e−1 (4.80e−3) =	1.7551e−1 (9.87e−2) −	1.3804e−1 (1.19e−1) −	2.6904e−1 (2.47e−2)
MW8	mean(std)	0.0000e+0 (0.00e+0) −	1.0865e−1 (1.15e−1) −	2.2645e−1 (3.82e−3) −	1.3388e−1 (9.84e−2) −	3.3319e−2 (7.38e−2) −	2.6421e−1 (2.53e−2)
MW9	mean(std)	1.5227e−1 (7.34e−2) −	1.2433e−1 (4.26e−2) −	4.1484e−1 (5.25e−2) =	2.9983e−1 (3.19e−2) −	8.9580e−2 (4.27e−3) −	**4.4110e−1 (4.15e−2)**
MW10	mean(std)	1.4825e−1 (1.08e−1) −	8.4735e−2 (1.90e−2) −	5.3747e−1 (1.13e−1) =	4.1811e−1 (9.10e−2) −	8.8403e−2 (5.09e−2) −	**6.0896e−1 (3.41e−2)**
MW11	mean(std)	1.9080e−1 (1.94e−2) −	2.3385e−1 (7.87e−2) −	5.0837e−1 (1.83e−1) =	5.6424e−1 (8.25e−2) −	2.0359e−1 (6.99e−2) −	6.6241e−1 (3.49e−2)
MW12	mean(std)	3.0653e−1 (5.46e−2) −	2.3748e−1 (9.92e−2) −	5.0741e−1 (4.02e−2) −	4.6849e−1 (3.74e−2) −	1.9870e−1 (4.51e−2) −	5.6178e−1 (2.70e−2)
MW13	mean(std)	4.1848e−4 (4.43e−5) −	1.2957e−4 (1.43e−4) −	4.7065e−1 (2.72e−2) =	5.4468e−1 (2.66e−3) +	4.3357e−4 (1.64e−5) −	**4.3387e−1 (5.20e−2)**
MW14	mean(std)	9.6277e−4 (2.84e−5) −	3.6181e−4 (2.74e−4) −	5.1899e−1 (1.18e−2) +	5.4552e−1 (1.62e−3) +	9.5614e−4 (6.53e−5) −	**4.6412e−1 (4.81e−2)**

续表

LIRCMOP1	mean(std)	4.7765e−1 (1.73e−2) −	4.8870e−1 (3.68e−3) −	3.8396e−1 (1.46e−1) −	4.8888e−1 (3.89e−4) −	4.7715e−1 (1.49e−2) −	**4.8933e−1 (2.13e−4)**
LIRCMOP2	mean(std)	5.4847e−1 (1.40e−2) −	5.3971e−1 (1.59e−2) −	3.8497e−1 (1.63e−1) −	5.5819e−1 (1.12e−2) −	5.4586e−1 (1.15e−2) −	5.8067e−1 (6.15e−4)
LIRCMOP3	mean(std)	5.4268e−1 (8.11e−4) −	5.4309e−1 (6.70e−4) −	5.4251e−1 (7.99e−4) −	5.4412e−1 (5.62e−4) +	5.4396e−1 (9.85e−4) =	**5.4363e−1 (4.09e−4)**
LIRCMOP4	mean(std)	8.3998e−1 (2.17e−3) +	8.2180e−1 (2.26e−3) −	7.9482e−1 (1.62e−2) −	8.3812e−1 (2.30e−4) +	8.4129e−1 (2.12e−4) +	**8.3261e−1 (1.72e−3)**
LIRCMOP5	mean(std)	3.2222e−1 (1.54e−3) =	2.2488e−1 (1.16e−1) −	2.5640e−1 (8.81e−2) −	3.1507e−1 (1.90e−3) −	2.0201e−1 (1.09e−1) −	3.2329e−1 (3.93e−4)
LIRCMOP6	mean(std)	2.9598e−1 (3.49e−2) −	2.9831e−1 (1.04e−2) −	1.3223e−1 (1.00e−1) −	3.1749e−1 (1.01e−2) −	2.7833e−1 (4.49e−2) −	**3.2820e−1 (1.65e−4)**
LIRCMOP7	mean(std)	4.1082e−1 (4.46e−4) =	3.7821e−1 (7.04e−2) =	4.1115e−1 (7.97e−4) =	4.0815e−1 (9.01e−4) −	4.1079e−1 (8.04e−4) =	**4.1106e−1 (5.75e−4)**
LIRCMOP8	mean(std)	5.2619e−1 (1.21e−2) =	4.8911e−1 (2.07e−2) −	3.5880e−1 (1.06e−1) −	5.1127e−1 (2.14e−2) =	5.1583e−1 (2.17e−2) =	**5.2512e−1 (8.95e−3)**
LIRCMOP9	mean(std)	3.8387e−1 (4.08e−3) −	3.8623e−1 (6.04e−3) −	1.7613e−1 (1.66e−1) −	3.9079e−1 (2.21e−3) −	2.7869e−1 (1.52e−1) −	**3.9612e−1 (3.30e−3)**
LIRCMOP10	mean(std)	4.0317e−1 (3.71e−2) −	3.7787e−1 (4.87e−2) −	2.5533e−1 (1.05e−1) −	4.3529e−1 (1.12e−2) −	3.4175e−1 (8.96e−2) −	**4.5381e−1 (5.95e−4)**
LIRCMOP11	mean(std)	3.5163e−1 (8.33e−2) −	3.8112e−1 (8.59e−2) =	4.4718e−1 (2.00e−4) +	4.4099e−1 (1.13e−3) −	2.7760e−1 (1.71e−2) −	**4.4618e−1 (3.37e−4)**
LIRCMOP12	mean(std)	6.0406e−1 (8.58e−4) +	6.0393e−1 (5.76e−4) +	3.5750e−1 (2.63e−1) −	6.0044e−1 (4.13e−4) −	3.9149e−1 (2.73e−1) −	**6.0334e−1 (3.73e−4)**
LIRCMOP13	mean(std)	4.2828e−1 (2.45e−2) −	4.2774e−1 (1.51e−2) −	2.4281e−1 (1.29e−1) −	4.6020e−1 (1.23e−2) −	3.7056e−1 (7.65e−2) −	4.7144e−1 (5.29e−3)
LIRCMOP14	mean(std)	4.4019e−1 (3.36e−3) −	4.5277e−1 (4.26e−3) −	4.3607e−1 (1.24e−2) −	4.5944e−1 (1.17e−2) −	4.5910e−1 (7.47e−3) −	4.6720e−1 (2.65e−3)

Table 3. Comparison of Feasibility Rate metrics of various algorithms on test problems
表 3. 各算法在测试问题上的 Feasibility Rate 指标对比

Problem		CMOEAD	NSGA-II	PPS	CTAEA	NSGA-III	EMFEACD
MW1	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.6000e-2 (1.65e-2) -	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW2	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	3.0000e-2 (1.63e-2) -	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW3	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.1000e-2 (1.45e-2) -	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW4	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	8.0000e-3 (9.19e-3) -	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW5	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW6	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW7	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**

续表

MW8	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0)	1.0000e+0 (0.00e+0)=	**1.0000e+0 (0.00e+0)**
MW9	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW10	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW11	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW12	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW13	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
MW14	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP1	mean(std)	8.0000e-1 (4.22e-1) =	9.0000e-1 (3.16e-1) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	9.0000e-1 (3.16e-1) -	**1.0000e+0 (0.00e+0)**
LIRCMOP2	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP3	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP4	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP5	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP6	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP7	mean(std)	1.0000e+0 (0.00e+0) +	1.0000e+0 (0.00e+0) +	1.0000e+0 (0.00e+0) +	1.0000e+0 (0.00e+0) +	1.0000e+0 (0.00e+0) +	9.8800e-1 (1.55e-2)
LIRCMOP8	mean(std)	1.0000e+0 (0.00e+0) +	1.0000e+0 (0.00e+0) +	1.0000e+0 (0.00e+0) +	1.0000e+0 (0.00e+0) +	1.0000e+0 (0.00e+0) +	9.8800e-1 (1.81e-2)
LIRCMOP9	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP10	mean(std)	1.0000e+0 (0.00e+0) =	9.0000e-1 (3.16e-1) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP11	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP12	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP13	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**
LIRCMOP14	mean(std)	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	**1.0000e+0 (0.00e+0)**

Table 4. Comparison of runtime metrics of various algorithms on test problems
表 4. 各算法在测试问题上的 runtime 指标对比

Problem		CMOEAD	NSGA-II	PPS	CTAEA	NSGA-III	EMFEACD
MW1	mean(std)	1.1744e+1 (6.10e−1) −	1.3935e+0 (1.94e−2) +	6.3036e+0 (2.88e−1) −	3.7770e+1 (2.46e+0) −	1.8542e+0 (1.31e−1) +	4.1010e+0 (6.57e−2)
MW2	mean(std)	1.1691e+1 (6.41e−2) −	1.3821e+0 (2.28e−2) +	6.0760e+0 (3.73e−2) −	3.6687e+1 (3.10e+0) −	1.5075e+0 (5.24e−2) +	4.2040e+0 (4.73e−2)
MW3	mean(std)	1.1740e+1 (6.23e−2) −	1.3828e+0 (1.62e−2) +	6.1441e+0 (3.38e−2) −	3.6009e+1 (1.75e+0) −	1.6154e+0 (1.38e−1) +	4.1571e+0 (6.94e−2)
MW4	mean(std)	1.1848e+1 (5.51e−2) −	1.3998e+0 (2.31e−2) +	6.1495e+0 (3.90e−2) −	4.1047e+1 (3.15e+0) −	1.9647e+0 (4.67e−2) +	4.1478e+0 (7.02e−2)
MW5	mean(std)	1.1641e+1 (6.19e−2) −	1.3471e+0 (1.08e−2) +	6.1692e+0 (1.22e−1) −	5.3627e+1 (8.38e−1) −	2.0873e+0 (5.30e−2) +	4.5011e+0 (8.58e−2)
MW6	mean(std)	1.1617e+1 (2.91e−2) −	1.3338e+0 (1.57e−2) +	6.0822e+0 (5.77e−2) −	5.2432e+1 (7.58e−1) −	2.0649e+0 (4.16e−2) +	4.5676e+0 (1.09e−1)
MW7	mean(std)	1.1750e+1 (9.83e−2) −	1.4006e+0 (1.46e−1) +	6.2094e+0 (2.48e−2) −	5.4501e+1 (2.74e+0) −	2.0620e+0 (3.68e−2) +	4.3381e+0 (7.71e−2)
MW8	mean(std)	1.1709e+1 (1.06e−1) −	7.2880e−1 (8.94e−3) +	6.2050e+0 (2.79e−2) −	5.3752e+1 (1.02e+0) −	2.1549e+0 (5.57e−2) +	2.5023e+0 (1.56e−1)
MW9	mean(std)	1.1631e+1 (4.82e−2) −	7.3468e−1 (3.20e−3) +	6.1141e+0 (3.77e−2) −	5.0016e+1 (1.48e+0) −	2.0958e+0 (5.72e−2) +	2.5600e+0 (1.19e−1)
MW10	mean(std)	1.2224e+1 (1.29e−1) −	7.6871e−1 (1.26e−2) +	6.3621e+0 (3.80e−2) −	4.7094e+1 (1.86e+0) −	1.9533e+0 (2.07e−1) +	2.6763e+0 (9.06e−2)
MW11	mean(std)	1.2079e+1 (1.70e−1) −	7.5931e−1 (2.49e−3) +	6.3543e+0 (9.09e−2) −	4.7105e+1 (1.89e+0) −	1.9802e+0 (5.49e−2) +	2.7639e+0 (3.95e−1)
MW12	mean(std)	1.2044e+1 (6.66e−2) −	7.3632e−1 (8.78e−3) +	6.1690e+0 (1.05e−1) −	4.7538e+1 (1.24e+0) −	1.9296e+0 (5.32e−2) +	2.4591e+0 (9.57e−2)
MW13	mean(std)	1.1623e+1 (1.00e−1) −	7.6315e−1 (7.03e−3) +	6.1659e+0 (3.88e−2) −	6.4152e+1 (1.41e+0) −	1.9601e+0 (4.40e−2) +	5.1879e+0 (1.09e−1)
MW14	mean(std)	1.1683e+1 (4.40e−2) −	7.5786e−1 (6.68e−3) +	6.2382e+0 (4.13e−2) −	5.4701e+1 (7.76e−1) −	1.9179e+0 (4.56e−2) +	4.5986e+0 (1.19e−1)
LIRCMOP1	mean(std)	1.2004e+1 (7.52e−2) −	1.7950e+0 (1.84e−1) +	6.4051e+0 (1.21e−1) +	5.5531e+1 (3.18e+0) −	2.5594e+0 (8.55e−1) +	8.8133e+0 (2.71e−1)
LIRCMOP2	mean(std)	1.1845e+1 (6.51e−2) −	1.3073e+0 (1.24e−2) +	6.3662e+0 (6.61e−2) +	6.0798e+1 (5.42e−1) −	2.0686e+0 (4.83e−2) +	8.4273e+0 (1.30e−1)
LIRCMOP3	mean(std)	1.1592e+1 (7.54e−2) −	1.2629e+0 (1.06e−2) +	6.3242e+0 (5.57e−2) −	5.3670e+1 (8.46e−1) −	2.0324e+0 (4.77e−2) +	5.0474e+0 (8.31e−2)
LIRCMOP4	mean(std)	1.2217e+1 (5.27e−2) −	1.6646e+0 (1.57e−1) +	6.9398e+0 (4.64e−1) +	6.2778e+1 (8.25e−1) −	2.5919e+0 (4.11e−1) +	1.0180e+1 (7.78e−2)
LIRCMOP5	mean(std)	1.2306e+1 (9.76e−2) −	1.6238e+0 (4.02e−1) +	6.7544e+0 (5.10e−2) −	4.5429e+1 (7.00e−1) −	2.3543e+0 (5.91e−1) +	5.3262e+0 (9.06e−2)
LIRCMOP6	mean(std)	1.1816e+1 (8.84e−2) −	1.6782e+0 (3.28e−2) +	6.3990e+0 (2.76e−2) +	5.8191e+1 (1.92e+0) −	2.0770e+0 (6.84e−2) +	7.1956e+0 (1.10e−1)

续表

LIRCMOP7	mean(std)	1.1605e+1 (9.30e−2) −	1.5992e+0 (7.65e−2) +	6.3075e+0 (6.58e−2) −	4.7299e+1 (4.67e−1) −	2.0514e+0 (8.50e−2) +	6.0560e+0 (6.39e−2)
LIRCMOP8	mean(std)	1.2132e+1 (1.81e−1) −	1.4865e+0 (2.25e−2) +	6.6919e+0 (1.62e−1) +	7.0192e+1 (1.30e+0) −	2.2052e+0 (9.11e−2) +	9.2510e+0 (1.28e−1)
LIRCMOP9	mean(std)	1.2514e+1 (2.49e−1) −	1.4314e+0 (1.56e−1) +	6.5375e+0 (9.42e−2) −	4.5406e+1 (3.29e+0) −	1.7041e+0 (9.24e−2) +	4.5480e+0 (7.21e−2)
LIRCMOP10	mean(std)	1.2031e+1 (9.50e−2) −	1.6833e+0 (2.34e−1) +	6.4836e+0 (1.64e−1) +	6.0998e+1 (1.39e+0) −	2.2756e+0 (7.66e−2) +	7.8089e+0 (2.92e−1)
LIRCMOP11	mean(std)	1.1630e+1 (9.26e−2) −	1.4340e+0 (4.70e−2) +	6.2992e+0 (2.98e−2) −	4.2987e+1 (2.29e−1) −	1.9372e+0 (4.44e−2) +	5.3763e+0 (3.85e−1)
LIRCMOP12	mean(std)	1.2083e+1 (6.81e−2) −	1.7790e+0 (7.78e−2) +	6.7251e+0 (5.85e−2) −	4.9103e+1 (5.31e−1) −	2.1572e+0 (8.38e−2) +	5.8363e+0 (2.93e−1)
LIRCMOP13	mean(std)	1.1879e+1 (6.88e−2) −	1.3763e+0 (1.98e−2) +	6.6146e+0 (5.40e−2) −	5.7111e+1 (5.25e−1) −	2.1741e+0 (9.70e−2) +	5.8866e+0 (6.70e−2)
LIRCMOP14	mean(std)	1.2481e+1 (3.39e−2) −	1.4456e+0 (2.38e−2) +	7.1657e+0 (5.58e−2) +	6.9519e+1 (2.22e+0) −	2.1805e+0 (5.92e−2) +	7.4363e+0 (1.72e−1)

4.2. 参数敏感性分析

为评估 EMFEACD 算法对关键参数的鲁棒性，本节在 MW1 问题上测试自适应学习率 α 和知识迁移概率 β 的影响，两参数分别取值{0.1,0.2,0.3}，保持其余设置不变，每组独立运行 30 次，结果见表 5。实验表明：当 $\alpha=0.2,\beta=0.2$ 时算法性能最优，IGD 达 1.7770×10^{-3}，HV 为 4.8933×10^{-1}，且标准差最小；β 参数较为敏感，固定 $\alpha=0.2$ 时，β 降至 0.1 会导致 IGD 恶化近 3 倍，说明适中的知识迁移概率对归档协同至关重要；相比之下 α 较为稳定，在[0.1, 0.3]范围内变化时性能波动较小，但 $\alpha=0.3$ 时性能略有下降。综上，推荐采用 $\alpha=0.2,\beta=0.2$ 作为默认参数配置出现轻微波动。

Table 5. Sensitivity analysis results of parameters α and η on MW1
表 5. 参数 α 和 η 在 MW1 问题上的敏感性分析实验结果

实验组	α	β	IGD	HV
组一	0.2	0.2	**1.7770e−3 (7.16e−5)**	**4.8933e−1 (2.13e−4)**
组二	0.1	0.2	2.4077e−3 (1.94e−3)	4.8816e−1 (3.56e−3)
组三	0.3	0.2	3.9758e−3 (5.28e−3)	4.8554e−1 (8.73e−3)
组四	0.2	0.1	6.7128e−3 (7.80e−3)	4.8098e−1 (1.29e−2)
组五	0.3	0.1	8.5132e−3 (8.73e−3)	4.7793e−1 (1.44e−2)

4.3. 自适应权重变化曲线分析

由图 2 的算子权重演化曲线可见，EMFEACD 算法表现出良好的自适应特征。进化初期，各算子等权重运行以广泛探索搜索空间；随代数增加，各算子根据贡献度反馈产生分化，其中 DA 算子因在处理约束冲突和维持多样性上的显著贡献，权重最终上升至约 0.49。这种动态调整机制验证了算法能有效根据进化阶段需求平衡探索(Exploration)与开发(Exploitation)强度，是算法保持高性能的关键。

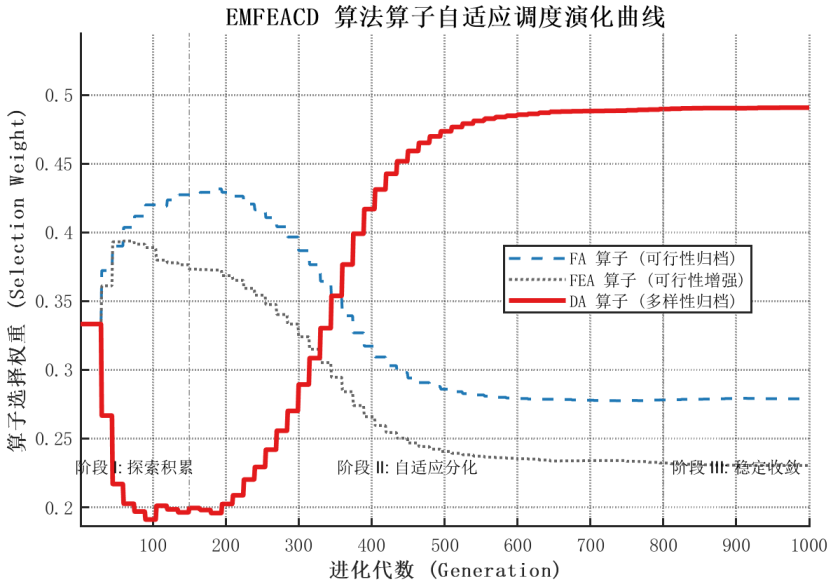

Figure 2. Adaptive weight variation
图 2. 自适应权重变化

5. 结论

本文针对约束多目标优化问题提出 EMFEACD 算法，通过多归档协同、技能因子自适应学习及混合算子策略，实现了收敛性、可行性与多样性的协调优化。三类功能归档的动态知识迁移有效提升了搜索效率与解集质量；DE 与 GA 的自适应组合使算法在探索与开发间达到良好平衡。在 MW 与 LIR-CMOP 测试集上的实验证实，EMFEACD 在 IGD 指标上显著优于 NSGA-II、CTAEA、PPS 与 CMOEAD 等主流算法，表现为更快的前沿逼近速度、更均匀的目标空间分布以及更广的覆盖范围。算法在不同测试问题与随机种子下保持稳定低方差，充分验证了多归档协同与技能因子机制的有效性。同时，EMFEACD 在复杂约束下维持高可行解比例，展现出良好的工程实用性。

综上所述，EMFEACD 为复杂约束多目标优化提供了一种高效、稳健且适用性广泛的求解范式，也为该领域算法设计提供了新思路。未来工作可向多任务优化、动态环境优化及高维多目标领域拓展，并进一步在能源系统、物流调度、智能制造等实际场景中验证应用价值。通过持续优化归档管理与算子设计，EMFEACD 有望在保持收敛–多样性平衡的同时，进一步提升对复杂约束问题的求解能力，为进化优化研究与应用提供有力支撑。

参考文献

[1] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002) A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**, 182-197. https://doi.org/10.1109/4235.996017

[2] Qingfu Zhang, and Hui Li, (2007) MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, **11**, 712-731. https://doi.org/10.1109/tevc.2007.892759

[3] Deb, K. (2000) An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, **186**, 311-338. https://doi.org/10.1016/s0045-7825(99)00389-8

[4] Fan Z, Li W, Cai R, *et al.* (2019) An Improved Push and Pull Search Framework for Constrained Multi-Objective Optimization. *Swarm and Evolutionary Computation*, **44**, 218-231.

[5] Ma, Z. and Wang, Y. (2019) Evolutionary Constrained Multiobjective Optimization: Test Suite Construction and Performance Comparisons. *IEEE Transactions on Evolutionary Computation*, **23**, 972-986. https://doi.org/10.1109/tevc.2019.2896967

[6] Fan, Z., Li, W., Cai, R., *et al.* (2019) LIR-CMOP: Large-Scale, Irregular, and Robust Constrained Multi-Objective Optimization. *IEEE Transactions on Evolutionary Computation*, **23**, 51-64.

[7] 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

[8] Li, K., Chen, R., Fu, G. and Yao, X. (2019) Two-Archive Evolutionary Algorithm for Constrained Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, **23**, 303-315. https://doi.org/10.1109/tevc.2018.2855411

[9] Gupta, A., Ong, Y. and Feng, L. (2016) Multifactorial Evolution: Toward Evolutionary Multitasking. *IEEE Transactions on Evolutionary Computation*, **20**, 343-357. https://doi.org/10.1109/tevc.2015.2458037

[10] Ming, F., Gong, W. and Wang, L. (2024) A Review of Constrained Multi-Objective Evolutionary Algorithms: Strategies and Applications. *Knowledge-Based Systems*, **284**, Article ID: 111244.

[11] Jiao, L., Liu, Z., Shang, R., *et al.* (2024) Dual-Stage Cooperative Coevolution for Constrained Multi-Objective Optimization. *IEEE Transactions on Systems*, *Man*, *and Cybernetics*: *Systems*, **54**, 1560-1572.

[12] Chen, R., Liu, J. and Zhang, Y. (2024) Adaptive Operator Selection based on Multi-objective Evolutionary Algorithm for Con-Strained Optimization. *Swarm and Evolutionary Computation*, **85**, Article ID: 101482.

[13] Wang, X., Tian, Y., Zhang, X., *et al.* (2024) Archive-Based Knowledge Transfer for Complex Constrained Optimization. *IEEE Transactions on Evolutionary Computation*, **28**, 1020-1034.

[14] Zhao, H., Li, X. and Liu, M. (2024) Multi-Archive Evolutionary Algorithm with Dynamic Resource Allocation for Constrained Multi-Objective Optimization. *Expert Systems with Applications*, **238**, Article ID:122110.

[15] Liu, Z., Wang, J. and Zheng, X. (2025) Knowledge Transfer in Evolutionary Multi-Tasking for Constrained Multi-Objective Optimization. *Information Sciences*, **691**, Article ID: 121543.