基于聚类的求解多模态多目标优化问题的差分进化算法

梁静，乔康佳，岳才桐，余坤杰，曲博阳，徐若浩，李志萌，胡毅

Abstract

多目标优化问题（MOP）是在多个冲突的目标中找到一组非支配的解。目前在多目标优化方面大多数都是在多任务、约束、大规模等上进行的研究，但是在求解多模态多目标优化问题（MMOP）方面还没有很深入地研究。原论文设计了一种基于聚类的特殊拥挤距离（CSCD）方法来计算决策空间和目标空间的综合拥挤程度。随后，引入基于距离的精英选择机制（DBESM）来确定不同个体的学习样本。因此，本文改进了原论文基于 k-means 聚类的差分进化算法，利用离差平方和来创建树的方法进行聚类，从而在决策空间和目标空间中获得均匀分布的解。最后，在四个评价指标上均有 7 个及以上的测试集比原论文效果好。

关键词：多模态多目标优化；聚类；多样性保持机制

1 引言

多模态多目标优化问题简称 MMOP，已成为一种流行的问题类型。MMOP 的定义如下：

$$\begin{cases} \min \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T, \\ \text{s.t.} \quad \mathbf{x} = (x_1, \ldots, x_D)^T \in S \end{cases} \tag{1}$$

其中 m 是目标数量，$F(x)$ 为 m 维目标向量，x 是由 n 个决策变量 x_i 组成的 D 维决策向量。当 $\forall i = 1, 2, \ldots, m$，$f_i(\mathbf{x_a}) \leqslant f_i(\mathbf{x_b})$，当 $\exists j = 1, 2, \ldots, m$，$f_j(\mathbf{x_a}) < f_j(\mathbf{x_b})$，解 x_a 支配着另一个解 x_b，而帕累托最优解是不受任何其他任何解支配的。

MMOP 需要同时保持决策空间和目标空间的多样性，且一个 PF 对应多个等价的 PS，如图 1所示。决策向量构成决策空间，而决策向量是由决策变量组成。PS（Pareto Set）是在决策空间的非支配解，即帕累托最优解；PF（Pareto Front）是决策向量在目标空间中的映射向量，即帕累托前沿，换句话说就是在目标空间中最优解的集合。在处理 MOP 时，多目标进化算法（MOEA）由于能够在一次运行中找到多个解而被人们广泛使用，然而大多数 MOEA 只注重在目标空间中寻找 PF，而忽略了决策空间的信息。再加上在决策空间中帕累托最优子集可能具有不同的形状和位置，很难将个体均匀地收敛到每个帕累托最优子区域，因此 MMOP 的求解显得并不是那么容易。

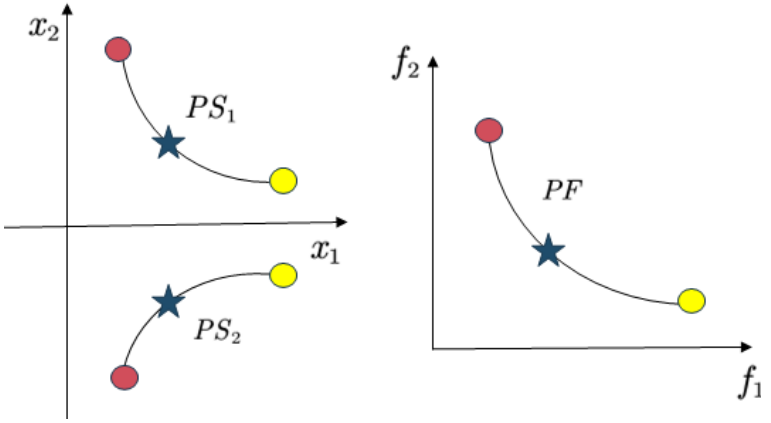

Figure 1: 多模态多目标优化问题图

MMOP 与传统的 MOP 有很大不同，MMOP 要求同时满足三个条件 [6]：1) 收敛性良好；2) 在目标空间中分布均匀；3) 在决策空间中分布均匀。

为了解决 MMOP，Biswas 等人把小生境技术被纳入到 MOEA 中 [1]。而小生境技术的灵感来自于自然界中生物进化的方式，它们最初被引入是为了解决单目标多模态问题。这些技术旨在形成稳定的生态位，以防止解收敛到局部区域。有些小生境技术可以保持决策空间中种群的多样性，如拥挤 [19]、适应度共享 [22]、聚类 [2] 和清除 [5]。然而，在求解 MMOP 时，种群在决策空间和目标空间的分布都需要改进。为此，原论文提出了一种多模态多目标差分算法，该算法采用基于聚类的特殊拥挤度距离法和精英选择机制 (MMODE_CSCD)，将基于聚类的特殊拥挤距离 (CSCD) 嵌入到非支配排序方案 [3] 中，利用拥挤距离衡量种群中解的多样性程度。CSCD 算法通过将种群聚类为多个子种群，并利用拥挤距离从每个子种群中选择最优解，从而生成一个多样化较强的种群。而非支配排序方案首先将种群划分为多个非支配等级，在决策空间中通过 k-means 聚类算法将同一等级的所有解分为多个类。在此基础上，本文不采用 k-means 聚类，而是先利用距离找变量之间的相似性从而创建树，在树里把同一等级的所有解分成多个类。这样，同一类别的个体来自同一组帕累托最优解。拥挤度距离是计算出来的，并通过使用同一类的邻里关系分配给每个个体。最后，原论文还设计了基于距离的精英选择机制（DBESM）改善所有 PS 上的种群分布。原论文研究的主要贡献如下:

(1) 提出了一种基于聚类的特殊拥挤距离方法，在决策空间和目标空间中寻找邻域关系并计算各解的综合拥挤距离。然后在样本选择和环境选择过程中，将拥挤距离作为一个指标。

(2) 引入一种综合考虑群体多样性和收敛性的基于距离的精英选择机制，确定每个个体的学习样本。

本文的其余部分安排如下。第二节对传统的多目标进化算法（MOEA）、现有的三类多模态多目标优化算法（MMOA）和相关工作进行了全面的回顾分析，第三节详细介绍原文提出的方法，第四节介绍本文设计的算法细节，第五节分析实验的结果，第六节给出了详细的总结和未来的工作。

2 相关工作

2.1 传统的多目标进化算法

在 1993-1995 年，提出了许多不同的进化算法来解决多目标优化问题。其中，Zitzler 和 Thiele 在 SPEA 中提出了一个非支配概念的精英多准则进化算法 [30]。他们建议在每一代保持一个外部种群，存储从初始种群到目前为止发现的所有非支配解，这个外部种群参与了所有的遗传操作。Knowles 和 Corne[9] 提出了一种简单的 MOEA，它使用类似于 (1+1) 进化策略的单亲单后代进化算法。Rudolph 提出一个简单的精英 MOEA[21]，它将子代种群的非支配解与父代种群的非支配解进行比较，形成一个整体的非支配解集，成为下一次迭代的父代种群。如果该集合的大小不大于期望的种群大小，则包括子代种群中的其他个体。通过这种策略，他证明了该算法收敛到帕累托最优前沿。虽然这本身就是一项重要的成就，但该算法缺乏动力去完成第二个任务，即保持帕累托最优解的多样性。必须增加一个明确的多样性保持机制，使其更实用。

针对不同的 MMOP 提出了大量的多模态多目标优化算法（MMOA），大致可分为三类：基于基于帕累托的 MMOA、基于分解的 MMOA 和基于指标的 MMOA。

2.2 基于帕累托的 MMOA

基于帕累托的 MMOEA 采用帕累托支配原则选择收敛的解，然后增强决策空间的多样性来求解 MMOP。该类别有两个代表性的算法，它们分别是 Omni-optimizer[4] 和 DN-NSGA-II [13]。Omni-optimizer 是第一个结合基于决策和目标空间的拥挤距离和非支配排序来选择后代的 MMOA。DN-NSGA-II 是一种基于解距离的匹配选择方法。DNEA[17] 是一种双小生境 MMOA，并引入了两个共享函数。DNEA 和 DN-NSGA-II 都是基于 Omni-optimizer 的增强算法。MO_Ring_PSO_SCD[28] 采用环拓扑形成稳定的小生境，并采用一种新的拥挤距离计算方法来选择子代，以保持较高的种群多样性。此外，Liu 等人 [16] 提出了 TriMOEA-TA&R，使用两种压缩和重组策略，并设置了两个档案。随后，提出了一种收敛惩罚密度技术，称为 CPDEA 算法 [18]。MMOEA/DC[14] 可以准确地找到局部 PS。

2.3 基于分解的 MMOA

基于分解的方法采用将给定的 MMOP 分解为一组单目标子问题，集成了决策空间的多样性以寻找多模态最优解。这类 MMOA 通常通过分解技术 [29] 将待解决的问题分解为许多单目标子问题。在 [7] 中，每个子问题分配了 k 个解。在此基础上，设计了一个基于惩罚边界相交点的适应度评价指标进行子代筛选 [8]。然而，有点难确定参数 k。因此，它被 [26] 中的一个不敏感参数（邻域大小）所取代。Hu 和 Ishibuchi[7] 提出了一种基于分解的方法，将多样性维持机制集成到决策空间中。在 [24] 中，子问题的种群大小由该分解框架中的两个操作符（即 Addition 或 Deletion）动态更新。Pal 等人 [20] 采用基于图拉普拉斯算子聚类的方法对决策空间进行分解，并利用参考向量对目标空间进行分解。

2.4 基于指标的 MMOA

基于指标的方法采用性能指标来引导种群在决策空间和目标空间搜索，通过定义适应度指标来选择后代。但在计算适应度指标时，往往会涉及到一些敏感的参数，这就涉及到参数 k 的设置。受 Liu 等人 [15] 的启发，为 MMOP 设计了一个权重适应度指标。[25] 中的算法也是一种基于指标的 MMOA，它引导种群使用选定的性能指标在小生境中进行搜索。

尽管已经有几种不同的 MMOA，但它们都有一些共同的缺点需要克服。

1) 大多数使用特殊拥挤度距离（SCD）方法，但在某些特殊情况下 SCD 无法找到合理的邻域。

2) 改善在不同 PS 上的分布。

为了克服这些问题，本文提出了 MMODE_CSCD 方法，该方法主要有两种方案：基于聚类的特殊拥挤距离（CSCD）和基于距离的精英选择机制（DBESM）。CSCD 可以找到正确的邻居关系。DBESM 可以保持种群多样性，提高在所有 PS 上的种群分布的均匀性。

3 本文方法

3.1 本文方法概述

本文主要是用到了原论文提到的在每个非支配等级上继续研究，在每个非支配等级把个体划分成多个类别，在聚类的方法上进行改进，利用切比雪夫距离寻找变量之间的相似性，更好地把更多相似种群划分成多个类别。然后在每个类别中算特殊拥挤度距离（CSCD），最后根据 CSCD 的大小进行环境选择，选择出优秀的个体。

3.2 基于聚类的特殊拥挤距离

特殊拥挤距离 (Special crowding distance, SCD) 是 MMOP 的一种常用方法，它为在同一非支配等级上的个体进行排序提供了一个指标。在接下来的部分中，首先介绍了 SCD 方法及其缺陷，然后说明了 CSCD 方法是如何解决这一问题的，以图 2为例。

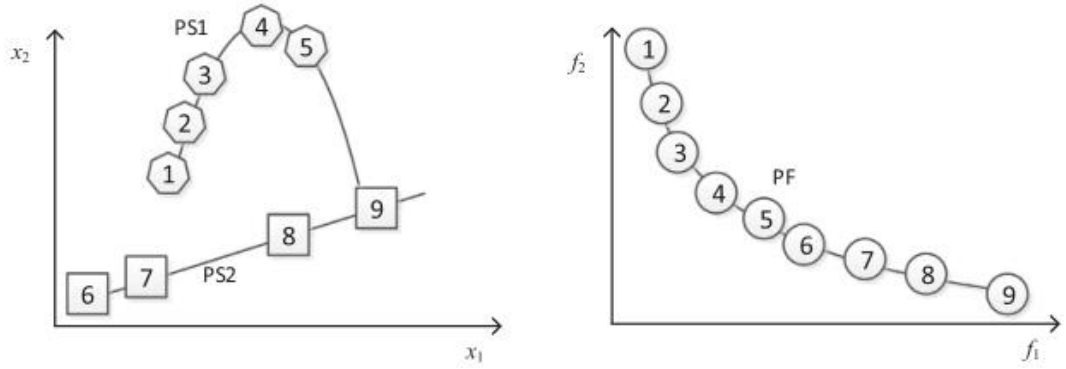

Figure 2: 决策空间和目标空间中解的分布图

具体来说，如果把两个 PS（即 PS1 和 PS2）看成一个整体来研究，从 x_1 坐标上看，编号为 8 的解左边相邻的解是 4，右边相邻的解是 5，因此根据以上信息，可通过下面式子计算出 $\mathrm{CD}_{8,x}$：

$$\mathrm{CD}_{8,x} = \left(\left|x_{4,1} - x_{5,1}\right| / \left|x_{6,1} - x_{9,1}\right|\right) + \left(\left|x_{7,2} - x_{9,2}\right| / \left|x_{6,2} - x_{4,2}\right|\right) \tag{2}$$

在目标空间中，分别从两个目标变量 f_1 和 f_2 上看，它们的排序相同，都是 (1, 2, 3, 4, 5, 6, 7, 8, 9)，因此 $\mathrm{CD}_{8,f}$ 可表示为：

$$\mathrm{CD}_{8,f} = \left(\left|x_{7,1} - x_{9,1}\right| / \left|x_{1,1} - x_{9,1}\right|\right) + \left(\left|x_{7,2} - x_{9,2}\right| / \left|x_{1,2} - x_{9,2}\right|\right) \tag{3}$$

最后，得到 SCD_8 的计算公式：

$$\mathrm{SCD}_8 = \begin{cases} \max\left(\mathrm{CD}_{8,x}, \mathrm{CD}_{8.f}\right), & \text{if } \mathrm{CD}_{8,x} > \mathrm{CD}_{avg,x} \text{ or } \mathrm{CD}_{8,f} > \mathrm{CD}_{avg.f} \\ \min\left(\mathrm{CD}_{8,x}, \mathrm{CD}_{8,f}\right), & \text{其他} \end{cases} \tag{4}$$

而如果单独在只一个 PS 上进行研究，在 PS2 中从横坐标看，编号为 8 的解左边相邻的解就是 7，右边是 9，可通过下面式子计算出 $\mathrm{CD}_{8,x}$、$\mathrm{CD}_{8,f}$ 和 CSCD_8：

$$\mathrm{CD}_{8,x} = \left(\left|x_{7,1} - x_{9,1}\right| / \left|x_{6,1} - x_{9,1}\right|\right) + \left(\left|x_{7,2} - x_{9,2}\right| / \left|x_{6,2} - x_{9,2}\right|\right) \tag{5}$$

$$\mathrm{CD}_{8,f} = \left(\left|x_{7,1} - x_{9.1}\right| / \left|x_{1,1} - x_{9,1}\right|\right) + \left(\left|x_{7,2} - x_{9,2}\right| / \left|x_{1,2} - x_{9,2}\right|\right) \tag{6}$$

$$\mathrm{CSCD}_8 = \begin{cases} \max\left(\mathrm{CD}_{8,x}, \mathrm{CD}_{8,f}\right), & \text{if } \mathrm{CD}_{8,x} > \mathrm{CD}_{avg,x} \text{ or } \mathrm{CD}_{8,f} > \mathrm{CD}_{avg.f} \\ \min\left(\mathrm{CD}_{8,x}, \mathrm{CD}_{8,f}\right), & \text{其他} \end{cases} \tag{7}$$

可以看到，用公式 (7) 比用公式 (4) 计算拥的挤度距离要大，即用 CSCD 方法会使得种群多样性增加，且个体的邻居只能从同一类中找到，因此种群可以均匀地分布，不会陷入局部最优，然后把 CSCD 作为综合拥挤度距离来衡量种群的多样性。

由于 CSCD 方法是在由非支配排序方案获得的每个非支配等级上实现的，因此将非支配排序方案和 CSCD 方法相结合的最终实现过程称为 Non-dominated_CSCD_sorting，其伪代码见 Procedure 1。在非支配排序方案中，种群被划分为多个非支配等级 $(F_1, F_2, \ldots, F_{num})$，其中 num 为非支配等级的总

数。经过非支配排序，得到第一个指标，即每个个体的非支配排序。然后，通过 k-means 算法将每个 F_j 分成多个类。在每个类中，每个解再根据公式 (5) - (7) 来计算 CSCD 值，此时 CSCD 作为第二个指标。然后，对具有相同非支配排序的解按其 CSCD 值降序排序。最后，输出根据这两个指标进行排序后的新种群。

Procedure 1 Non-dominated_CSCD_sorting

Input: 种群 P
Output: 排序后的种群 P^*
对 P 进行非支配排序，得到多个非支配等级 $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_{num}$ (num 为非支配等级的总个数)，j=(1, 2, …, num) 表示个体在 F_j 中的非支配等级;
for $j = 1 : num$ **do**
    $k_j = \left\lceil \frac{|\mathrm{F}_j|}{n} \right\rceil$ ▷ $|F_j|$ 表示 F_j 中个体的数量，n 是一个正整数
    用 k-means 算法将 F_j 分成多个类;
    利用同一类内的邻域关系计算每个个体的 CSCD 值;
end
根据种群 P 的非支配排序和 CSCD 值对种群 P 进行排序，得到新的种群 P^*.

3.3 DBESM

进行 Non-dominated_CSCD_sorting 后，得到一个经过按非支配和 CSCD 值排好序的种群。在 DBESM 中，每个解的样本是由非支配排序、CSCD 和一个按顺序排列的距离指示器决定的。Procedure 2 给出了 DBESM 的伪代码。

DBESM 包含三个主要步骤：

首先，对于每个 x_i^l（x_i^l 表示第 l 等级的第 i 个解），目标等级 (记为 F_a) 是通过从集合 $\{1, 2, \ldots, l-1\}$ 中随机选择一个数字来确定的。F_a 中的每个个体也许成为 x_i^l 的候选样本。当 a<l 时，样本的优秀性得到了保证。特别地，当 $l = 1$（等级为 1）时，由于没有更好的 F，a 将被设为 1。

其次，F_a 中 CSCD 值最大的前 100 * p (p 设为 0.1) 解被视为候选样本，这些候选样本被用来提供一些有希望的进化方向。

最后，根据距离指标确定最终样本。距离指示器指的是 x_i^l 和所有候选样本之间的距离，因此，候选样本接近 x_i^l 成为最后样本的概率很高。

那么，为什么使用距离指示器的原因说明如下。如果有几个个体属于 $PS_{x_i^l}$（即 x_i^l 定位在 PS 的地方），x_i^l 应该从 $PS_{x_i^l}$ 选择样本优先在 $PS_{x_i^l}$ 产生后代。因此，种群可以均匀地分布在所有的 PS 上。

Procedure 2 DBESM

Input: 排序后的种群 P^*, F_s 的个数 num, 每个支配层 F 中的解的个数 $|F_1|, |F_2|, \cdots, |F_{num}|$
Output: 每个解的样本
for $l = 1 : num$ **do**
    for $i = 1 : |F_l|$ **do**
        通过从集合 $\{1, 2, \ldots, l-1\}$ 中随机选择一个数字确定目标等级 F_a(如果 $l = 1$，则 a 设为 1);
        从 F_a 中最大的 CSCD 值选择前 100*p 个解作为候选样本;
        计算 x_i^l 和所有候选样本之间的距离作为距离指标;
        进行轮盘赌选择，根据距离指标从候选样本中确定最终样本;
    end
end

3.4 MMODE_CSCD 框架

Procedure 3 描述了 MMODE_CSCD 的框架。在搜索空间中生成 *NP* 个个体的初始种群 *P*，并对每个个体的所有目标值进行评估。接下来，进行 Non-dominated_CSCD_sorting 以获得排序后的种群 P^*、非支配排序和每个个体的 CSCD 值。然后，在 P^* 上进行 DBESM，以确定每个个体的样本。接着，对变异算子和交叉算子进行执行，生成子代种群 *OP*。通过合并 *P* 和 *OP* 得到临时种群 *POP*。然后，根据 Non-dominated_CSCD_sorting 对 *POP* 进行排序，以获得种群 POP^*。随后，通过环境选择从 POP^* 中选择一个新的种群 *P*。环境选择的过程如图 3所示。最后，如果 $G < G_{max}$，循环继续，否则，输出属于第一等级 F_1 的解。

Procedure 3 MMODE_CSCD 框架

Input: 种群大小 *NP*, 最大代数 G_{max}, 比例因子 *FF*, 交叉率 *Cr*
Output: *PF*, *PS*
G = 1;
生成 *NP* 个个体的初始种群 *P*，并对每个目标上的所有个体进行评估;
while $G < G_{max}$ **do**
    对 *P* 执行 *Non-dominated_CSCD_sorting*，获得一个排序的种群 *P*;* ▷ *Procedure 1*
    在 *P** 上执行 *DBESM*，为每个个体选择一个样本; ▷ *Procedure 2*
    交叉变异，得到子代种群 *OP;*
    通过合并 *P* 和 *OP*，对临时种群 *POP* 执行 *Non-dominated_CSCD_sorting*，获得一个排序的种群 *POP*;* ▷ *Procedure 1*
    在 *POP** 上进行环境选择，获得新的种群 *P;*
    G = *G* + *1;*
end

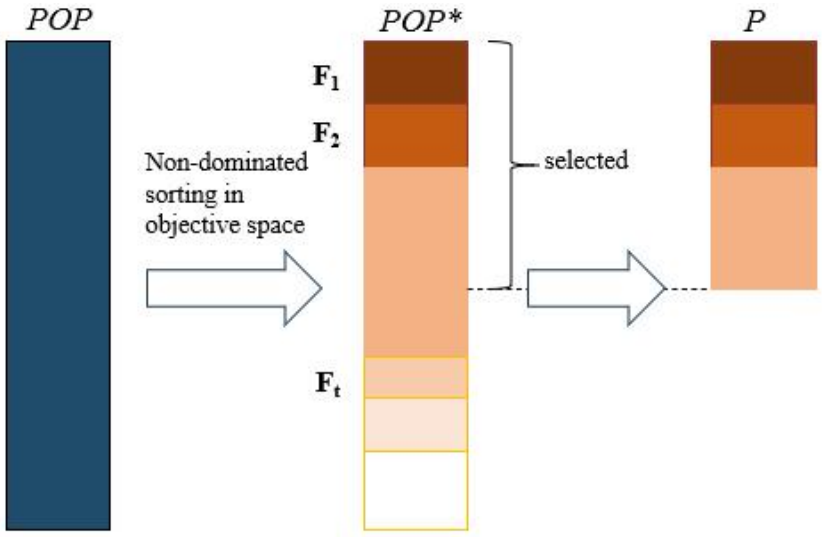

Figure 3: 环境选择过程图

4 复现细节

4.1 与已有开源代码对比

本文是在原论文已有开源代码的基础上进行改进的，主要是参考了源代码的整体架构，从而对算法细节进行了一些改进与创新，主要创新包括：数据预处理、聚类算子等，主要是在聚类算子上的改进，算法改进部分的伪代码见 Procedure 4。源代码中用的是 k-means 聚类的方法，其虽然原理简单，容易实现，但也易陷入到局部最优，且对于非凸数据集或类别规模差异太大的数据效果不好，因此本文想到用分步聚类的方法，先标准化决策空间的变量，从而进行数据预处理，根据变量之间的距离找

到它们之间的相似性，然后通过构建树的方式更好地把种群分成多个类别。最后跑 21 次实验取平均值，从而排除偶然性，得到的结果从四个评价指标来看均在超过 7 个问题集上比原论文要好。已有开源代码的网址为：http://www5.zzu.edu.cn/cilab/Code.htm。

Procedure 4 Non-dominated_CSCD_sorting_Tree_Clustering

Input: 种群 P
Output: 排序后的种群 P*
对 P 进行非支配排序，得到多个非支配等级 $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_{num}$ (num 为非支配等级的总个数) 以及每个非支配等级里的所有个体 ZhongjianPop，j=(1, 2, …, num) 表示个体在 F_j 中的非支配等级;
for $j = 1 : num$ **do**
    $k_j = \left\lceil \frac{|\mathrm{F}_j|}{n} \right\rceil$ ▷ $|F_j|$ 表示 F_j 中个体的数量，n 是一个正整数
    $X = \mathrm{Zscore}(ZhongjianPop)$
    $Y = \mathrm{Pdist}(X)$
    $Z = \mathrm{Linkage}(Y)$
    $Ind = \mathrm{Cluster}(Z, K)$
    利用同一类内的邻域关系计算每个个体的 CSCD 值;
end
根据种群 P 的非支配排序和 CSCD 值对种群 P 进行排序，得到新的种群 P*.

原论文里给出用 k-means 聚类的原因是简单、流行，所以本文尝试使用了其他聚类方法，用最短距离算法生成具有层次结构的聚类树，该方法作为一个指示器再次测量种群之间的距离，改进步骤如下：

1）Z-score 标准化，将原始数据均转换为无量纲化指标测评值，即各指标值都处于同一个数量级别上，可以进行综合测评分析。公式如下：

$$Z_i = (x_i - \mu) / \sigma \tag{8}$$

其中，μ 为均值，σ 为标准差。

2）然后利用切比雪夫距离寻找变量之间的相似性，利用离差平方和来创建树，然后在这个树里面构建聚类，公式如下：

$$dis = \sqrt{\sum_{k=1}^{n} (x_{1k} - x_{2k})^2} \tag{9}$$

其中，x 代表决策空间的变量。

3）再根据沃德方差最小化算法进行聚类，公式如下：

$$d(u, v) = \sqrt{\frac{|v| + |s|}{T} d(v, s)^2 + \frac{|v| + |t|}{T} d(v, t)^2 - \frac{|v|}{T} d(s, t)^2} \tag{10}$$

其中，u 是 s 和 t 组成的新的聚类，v 是森林中未使用的聚类。$T = |v| + |s| + |t|$，$|*|$ 是聚类簇中观测值的个数。

4.2 实验环境搭建

该算法在 MATLAB R2021b 环境中实现，实验平台环境为 Window 10 系统，CPU 为 8GB 处理器。在 MATLAB 中建立一个工程文件夹，用于存放代码和数据，以方便实验数据的管理和使用，最后的数据存在 “.mat” 文件里。

4.3 界面分析与使用说明

MATLAB 在主程序窗口中嵌套了多个分区，包括工作区、命令行窗口、路径选择和功能区。“Indicators” 文件夹下是评价指标的文件，“MM_testfunctions” 文件夹下是 22 个问题集，其中包括两个文件夹：functions 和 Reference_PSPF_data，“functions” 文件夹下存放的是 22 个问题集，“Reference_PSPF_data” 文件夹下存放的是全局 PS 和 PF、局部 PS 和 PF 以及参考 PS 和 PF，“数据” 文件夹下存放的是经过 21 次实验后得到的 4 个评价指标的平均值、标准差和方差，“图片” 文件夹下存放的是得到的 PS 和 PF 与真实 PS 和 PF 对比结果图。

4.4 创新点

在数据预处理阶段，本文加入了标准化操作，使得数据点在每维上具有相似的宽度，可以起到一定的增大数据分布范围的作用。接着，在计算拥挤度距离阶段，采用分布聚类的方法，利用距离找变量之间的相似性来创建树，从而更好地把每个非支配等级中的所有个体分成多个类别，在每一个类别中计算决策空间的拥挤度距离，以此来增加决策空间的多样性。

5 实验结果分析

本文数据集用的是 CEC 2019 多模态多目标优化基准套件 [11]，该基准测试套件包含 22 个不同特征的测试函数，并与 3 个最先进的多模态多目标进化算法进行了性能比较，包括 MMODE_CSCD[10]、MMOPIO[27] 和 MO_PSO_MM[12]，这些比较算法的参数设置与原始文献一致，可以系统地验证算法的性能。

在我们的实验中，种群大小 P 设置为 200，而最大迭代次数设置为 20000。将聚类类别 n 设为 10，为了进行公平的比较，对所有对比算法在每个多模态多目标进化问题上独立运行 20 次。图 4-9分别是在第 4、9、20 测试集得到的 PS 和 PF 图，红色代表我的算法得到的 PS，蓝色代表真实的 PS，可以看到种群能够很好地分布在帕累托最优解集和帕累托前沿上。

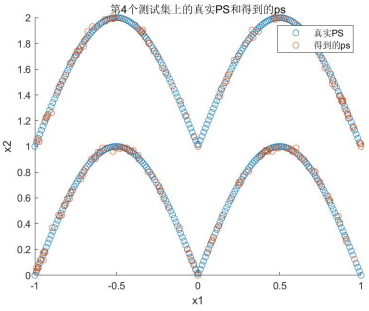

Figure 4: MMF4 问题集上的真实 PS 和得到的 ps

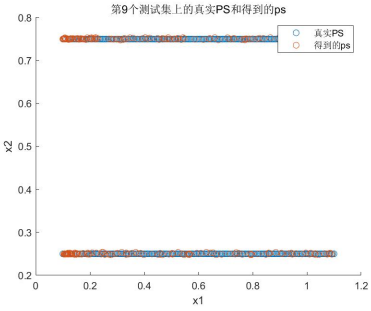

Figure 5: MMF9 问题集上的真实 PS 和得到的 ps

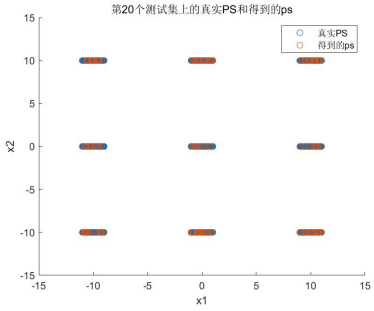

Figure 6: SYM-PART simple 问题集上的真实 PS 和得到的 ps

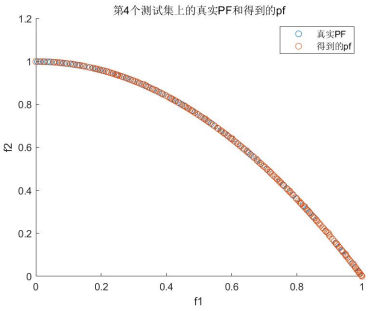

Figure 7: MMF4 问题集上的真实 PF 和得到的 pf

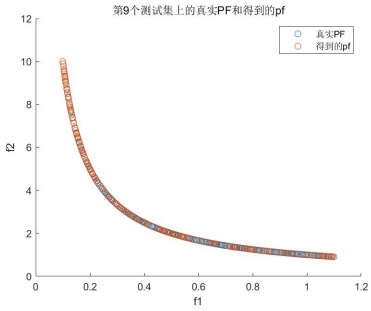

Figure 8: MMF9 问题集上的真实 PF 和得到的 pf

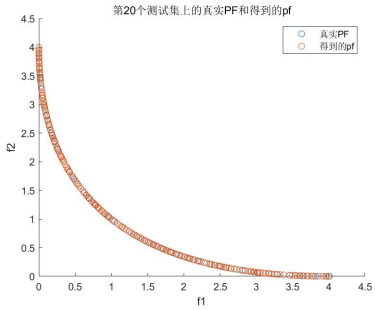

Figure 9: SYM-PART simple 问题集上的真实 PF 和得到的 pf

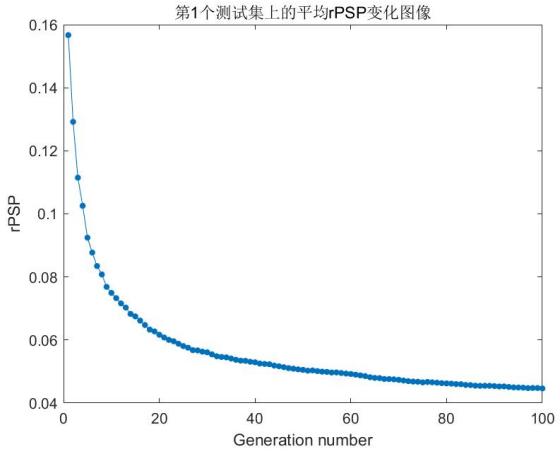

Figure 10: MMF1 上平均 rPSP 变化图

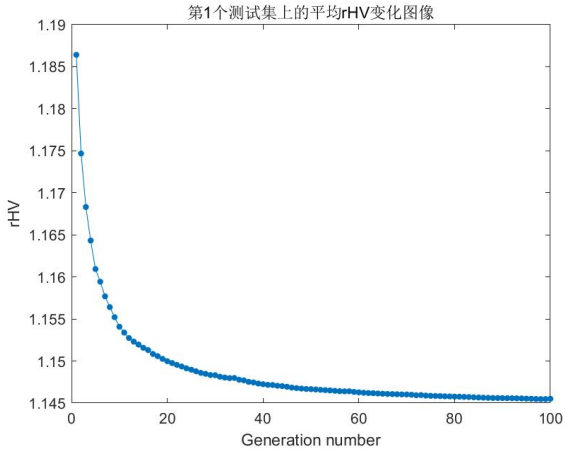

Figure 11: MMF1 上平均 rHV 变化图

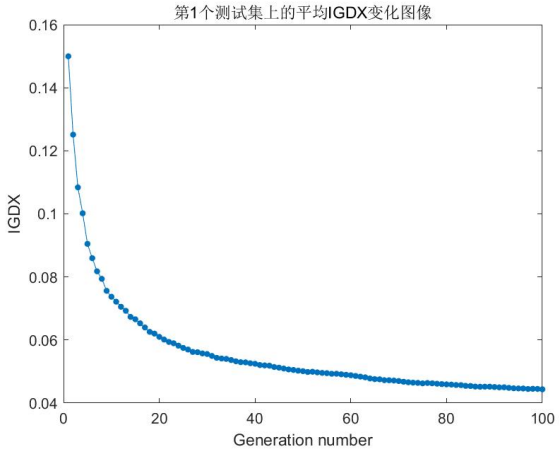

Figure 12: MMF1 上平均 IGDX 变化图

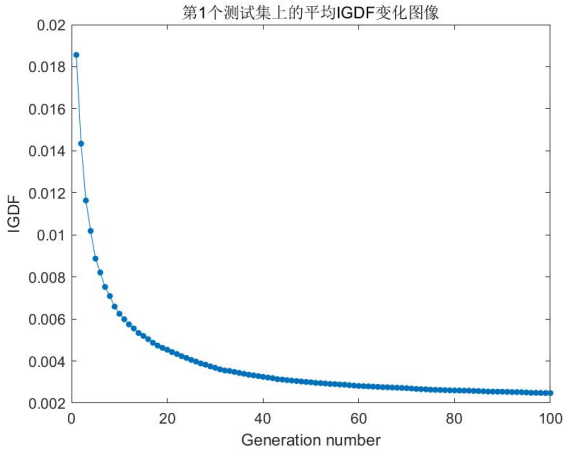

Figure 13: MMF1 上平均 IGDF 变化图

此外，该基准套件还提供了四个性能指标：帕累托集接近性 (Pareto Sets Proximity, PSP) 的倒数 (1/PSP, rPSP)、Hypervolume 的倒数 (1/HV, rHV)、决策空间中的倒代距离 IGD(IGDx) 和目标空间中的倒代距离 IGD(IGDf)。

rPSP 和 rHV 被用来代替 PSP 和 HV，因此对于这四个指标来说，数值越小意味着性能越好。其中 rPSP 和 IGDx 用于评价种群在决策空间中的分布，rHV 和 IGDf 用于评价种群在目标空间中的分布。虽然这四个指标的作用不同，但 rPSP 和 rHV 比 IGDx 和 IGDf 具有更全面的特征。

1）rPSP

rPSP 值的计算考虑了得到的最优解在真实 PS 上的覆盖率 (CR)，用来评估得到的 PS 与真实 PS 的相似程度，反映得到的解的多样性和收敛性，公式如下:

$$CR(\mathbf{X}) = \left(\prod_{i=1}^{D} \eta_i \right)^{\frac{1}{2D}} \tag{11}$$

$$RPSP(\mathbf{X}) = \frac{IGDX(\mathbf{X})}{CR(\mathbf{X})} \tag{12}$$

其中，$\eta_i = \left\{ \dfrac{\min\left(x_i^{*,\max}, x_i^{\max}\right) - \max\left(x_i^{*,\min}, x_i^{\min}\right)}{x_i^{*,\max} - x_i^{*,\min}} \right\}^2$，$x_i$ 和 x_i^* 表示第 i 个得到的解和真实 PS 上的参考解。

2）rHV

$$\mathrm{HV} = \mathrm{VOL}_{\mathbf{y} \in pf}\left(\mathbf{y}, \mathbf{R}^*\right) \tag{13}$$

其中 VOL 为勒贝格测度，$\mathbf{R}^*$ 为参考点。

3）IGDx

$$IGDx = \frac{\left(\sum_{j=1}^{|PS|} Dist_j^2\right)^{1/2}}{|PS|} \tag{14}$$

其中，$Dist_j$ 表示第 j 个采样参考决策向量与得到的最近决策向量之间的欧氏距离。

4）IGDf

它主要通过计算每个在真帕累托前沿面上的点到算法得到的点之间的最小距离和，来评价算法的收敛性能和分布性能，公式如下：

$$IGDf = \frac{\left(\sum_{i=1}^{|PF|} Dist_i^2\right)^{1/2}}{|PF|} \tag{15}$$

其中，$Dist_i$ 表示第 i 个采样参考目标向量与得到的最近目标向量之间的欧氏距离。

在此基础上，讨论了各种对比算法在各指标上的性能。表 1-4 记录了各种算法得到的 rPSP、rHV、IGDx、IGDf 的均值 (mean) 和标准差 (std) 的统计比较结果。图 10-图 13表示在 MMF1 测试集独立运行 21 次实验后得到的四个指标平均值变化情况图，可见指标可以降到很低。

表 1 列出了 3 个多模态多目标对比算法得到的 rPSP 值。提出的 MY_ALGORITHM 在 22 个多模态多目标进化问题中获得了明显更好的 rPSP 性能，在 7 个测试实例上明显比三个对比算法好，因此肯定比原论文提出的 MMODE_CSCD 算法好。在 Omni test 测试集上我设计的算法比原论文算法好，仅次于 MO_PSO_MM。

表 2 列出了对比多模态多目标进化算法得到的 rHV 值。与其他对比多模态多目标进化算法相比，MY_ALGORITHM 在 8 个多模态多目标进化问题中明显获得了更好的 rHV 性能。对于 MMF5, MMODE_CSCD 达到了最好的性能，而 MY_ALGORITHM 和 MO_PSO_MM 也不落后。事实上，所有算法都考虑了解在目标空间中的分布，因此它们的 rHV 值彼此接近。综上所述，本文设计的算法在目标空间中仍然取得了最好的性能。

表 3 列出了 3 个多模态多目标对比算法得到的 IGDx 值。本文提出的 MY_ALGORITHM 在 22 个多模态多目标进化问题中获得了明显更好的 IGDx 性能，在 7 个测试实例中胜出。尽管 MMODE_CSCD 在 MMF9、MMF11、MMF12、MMF14、MMF15、MMF15_a 和 SYM-PART simple 实例中获得了很好的 IGDx 值，但 MY_ALGORITHM 也比其他多模态多目标进化算法表现得更好。对于 MMF4、MMF6、MMF7, MMODE_CSCD 达到了最好的性能，而 MY_ALGORITHM 也不落后。

表 4 列出了对比多模态多目标进化算法得到的 IGDf 值。与其他对比多模态多目标进化算法相比，MY_ALGORITHM 在 12 个多模态多目标进化问题中获得了明显更好的 IGDf 性能。对于其余的多模态多目标进化算法, MMODE_CSCD 在 7 个实例中表现最好，MMOPIO 在 3 个实例上胜出。对于 MMF1、MMF5、MMF7、MMF9、MMF1_z 和 MMF1_e 实例，所提出的 MY_ALGORITHM 实现了良好的 IGDf 值，只是比 MMODE_CSCD 略差，反映出该算法在目标空间均有较好的收敛性和多样性。

实验结果表明，MY_ALGORITHM 可以处理凸、凹、断开和球面帕累托前沿等多种特征的多模态多目标进化问题。此外，这些多模态多目标进化问题实例有 2 到 27 个帕累托子区域。实验结果也证明了该方法能够在决策空间中找到多组分布良好且收敛良好的最优解集。

Table 1: 对比算法的指标 rPSP 值

问题集编号	MMODE_CSCD mean ± std	MMOPIO mean ± std	MO_PSO_MM mean ± std	MY_ALGORITHM mean ± std
MMF1	0.0414 ± 0.0014	0.0419 ± 0.0025	0.0418 ± 0.0025	0.0446 ± 0.0012
MMF2	0.0102 ± 0.0018	0.0124 ± 0.0036	0.0191 ± 0.0059	0.0384 ± 0.0067
MMF3	0.0085 ± 0.0018	0.0118 ± 0.0038	0.0137 ± 0.0018	0.0292 ± 0.0044
MMF4	0.0223 ± 0.0011	0.0288 ± 0.0042	0.0276 ± 0.0022	0.0239 ± 0.0014
MMF5	0.0721 ± 0.0034	0.0847 ± 0.0070	0.0751 ± 0.0036	0.0790 ± 0.0045
MMF6	0.0625 ± 0.0022	0.0722 ± 0.0045	0.0679 ± 0.0030	0.0670 ± 0.0027
MMF7	0.0221 ± 0.0014	0.0346 ± 0.0047	0.0321 ± 0.0037	0.0294 ± 0.0048
MMF8	0.0489 ± 0.0031	0.0634 ± 0.0132	0.0480 ± 0.0033	0.1943 ± 0.0623
MMF9	0.0057 ± 0.0003	0.0122 ± 0.0025	0.0098 ± 0.0017	**0.0051** ± 0.0007
MMF10	0.0370 ± 0.1063	0.0051 ± 0.0022	0.0019 ± 0.0003	0.1339 ± 0.1910
MMF11	0.0041 ± 0.0003	0.0072 ± 0.0017	0.0069 ± 0.0018	**0.0026** ± 0.0001
MMF12	0.0016 ± 0.0001	0.0022 ± 0.0005	0.0017 ± 0.0002	**0.0014** ± 0.0000
MMF13	0.0277 ± 0.0008	0.0335 ± 0.0022	0.0282 ± 0.0013	0.0382 ± 0.0082
MMF14	0.0624 ± 0.0016	0.0662 ± 0.0027	0.0631 ± 0.0023	**0.0552** ± 0.0019
MMF15	0.0504 ± 0.0019	0.0489 ± 0.0026	0.0475 ± 0.0018	**0.0415** ± 0.0023
MMF1_z	0.0288 ± 0.0009	0.0319 ± 0.0025	0.0308 ± 0.0018	0.0387 ± 0.0053
MMF1_e	0.3617 ± 0.1759	0.6117 ± 0.3614	0.3043 ± 0.1008	3.0504 ± 2.2153
MMF14_a	0.0741 ± 0.0026	0.0736 ± 0.0025	0.0726 ± 0.0024	0.0754 ± 0.0046
MMF15_a	0.0591 ± 0.0029	0.0548 ± 0.0023	0.0535 ± 0.0023	**0.0529** ± 0.0024
SYM-PART simple	0.0539 ± 0.0039	0.0708 ± 0.0105	0.0727 ± 0.0093	**0.0430** ± 0.0034
SYM-PART rotated	0.1075 ± 0.2512	0.1215 ± 0.2058	0.1257 ± 0.0104	0.3244 ± 0.3995
Omni-test	0.5636 ± 0.1177	0.6689 ± 0.1403	0.3133 ± 0.0788	0.3958 ± 0.1895

Table 2: 对比算法的指标 rHV 值

问题集编号	MMODE_CSCD mean ± std	MMOPIO mean ± std	MO_PSO_MM mean ± std	MY_ALGORITHM mean ± std
MMF1	1.1455 ± 0.0003	1.1481 ± 0.0009	1.1473 ± 0.0007	1.1455 ± 0.0005
MMF2	1.1497 ± 0.0009	1.1504 ± 0.0013	1.1602 ± 0.0022	1.1795 ± 0.0047
MMF3	1.1488 ± 0.0006	1.1495 ± 0.0017	1.1560 ± 0.0011	1.1695 ± 0.0035
MMF4	1.8529 ± 0.0009	1.8607 ± 0.0039	1.8659 ± 0.0049	**1.8511** ± 0.0003
MMF5	1.1454 ± 0.0003	1.1481 ± 0.0016	1.1470 ± 0.0007	1.1455 ± 0.0004
MMF6	1.1457 ± 0.0006	1.1474 ± 0.0006	1.1473 ± 0.0007	**1.1449** ± 0.0002
MMF7	1.1454 ± 0.0002	1.1515 ± 0.0016	1.1515 ± 0.0011	**1.1451** ± 0.0003
MMF8	2.3747 ± 0.0018	2.3799 ± 0.0030	2.3792 ± 0.0023	2.3768 ± 0.0041
MMF9	0.1032 ± 0.0000	0.1034 ± 0.0001	0.1035 ± 0.0001	**0.1032** ± 0.0000
MMF10	0.0781 ± 0.0018	0.0779 ± 0.0004	0.0778 ± 0.0001	0.0804 ± 0.0040
MMF11	0.0689 ± 0.0000	0.0690 ± 0.0000	0.0690 ± 0.0000	**0.0689** ± 0.0000
MMF12	0.6355 ± 0.0000	0.6358 ± 0.0002	0.6385 ± 0.0002	**0.6354** ± 0.0000
MMF13	0.0542 ± 0.0000	0.0543 ± 0.0000	0.0543 ± 0.0000	0.0542 ± 0.0000
MMF14	0.3581 ± 0.0151	0.3274 ± 0.0179	0.3189 ± 0.0280	0.3336 ± 0.0104
MMF15	0.2443 ± 0.0103	0.2257 ± 0.0103	0.2286 ± 0.0079	0.2384 ± 0.0099
MMF1_z	1.1455 ± 0.0002	1.1471 ± 0.0005	1.1469 ± 0.0003	**1.1453** ± 0.0004
MMF1_e	1.1711 ± 0.0109	1.1488 ± 0.0018	1.1526 ± 0.0017	1.1501 ± 0.0019
MMF14_a	0.3600 ± 0.0199	0.3079 ± 0.0134	0.3148 ± 0.0144	0.3356 ± 0.0191
MMF15_a	0.2498 ± 0.0096	0.2228 ± 0.0068	0.2273 ± 0.0092	0.2392 ± 0.0082
SYM-PART simple	0.0600 ± 0.0000	0.0601 ± 0.0000	0.0601 ± 0.0000	0.0600 ± 0.0000
SYM-PART rotated	0.0600 ± 0.0000	0.0601 ± 0.0000	0.0603 ± 0.0000	0.0601 ± 0.0000
Omni-test	0.0190 ± 0.0000	0.0190 ± 0.0000	0.0189 ± 0.0000	**0.0189** ± 0.0000

Table 3: 对比算法的指标 IGDx 值

问题集编号	MMODE_CSCD mean ± std	MMOPIO mean ± std	MO_PSO_MM mean ± std	MY_ALGORITHM mean ± std
MMF1	0.0412 ± 0.0013	0.0416 ± 0.0025	0.0416 ± 0.0024	0.0444 ± 0.0012
MMF2	0.0102 ± 0.0018	0.0120 ± 0.0034	0.0185 ± 0.0057	0.0371 ± 0.0058
MMF3	0.0085 ± 0.0018	0.0115 ± 0.0036	0.0133 ± 0.0018	0.0283 ± 0.0044
MMF4	0.0221 ± 0.0011	0.0286 ± 0.0041	0.0275 ± 0.0022	0.0237 ± 0.0014
MMF5	0.0718 ± 0.0034	0.0841 ± 0.0070	0.0749 ± 0.0036	0.0785 ± 0.0044
MMF6	0.0622 ± 0.0022	0.0717 ± 0.0043	0.0676 ± 0.0030	0.0666 ± 0.0027
MMF7	0.0220 ± 0.0013	0.0344 ± 0.0046	0.0320 ± 0.0036	0.0276 ± 0.0038
MMF8	0.0484 ± 0.0029	0.0626 ± 0.0121	0.0478 ± 0.0033	0.1849 ± 0.0548
MMF9	0.0057 ± 0.0003	0.0122 ± 0.0025	0.0098 ± 0.0017	**0.0051** ± 0.0007
MMF10	0.0368 ± 0.1063	0.0051 ± 0.0022	0.0019 ± 0.0003	0.1339 ± 0.1910
MMF11	0.0041 ± 0.0003	0.0072 ± 0.0017	0.0069 ± 0.0018	**0.0026** ± 0.0001
MMF12	0.0016 ± 0.0001	0.0022 ± 0.0005	0.0017 ± 0.0002	**0.0014** ± 0.0000
MMF13	0.0276 ± 0.0008	0.0333 ± 0.0022	0.0279 ± 0.0012	0.0346 ± 0.0063
MMF14	0.0624 ± 0.0016	0.0662 ± 0.0027	0.0631 ± 0.0023	**0.0552** ± 0.0019
MMF15	0.0504 ± 0.0019	0.0488 ± 0.0026	0.0475 ± 0.0018	**0.0415** ± 0.0023
MMF1_z	0.0286 ± 0.0009	0.0317 ± 0.0024	0.0306 ± 0.0018	0.0384 ± 0.0052
MMF1_e	0.3198 ± 0.1320	0.5041 ± 0.2247	0.2845 ± 0.0683	1.6276 ± 0.7212
MMF14_a	0.0740 ± 0.0026	0.0734 ± 0.0025	0.0724 ± 0.0024	0.0753 ± 0.0046
MMF15_a	0.0538 ± 0.0038	0.0707 ± 0.0105	0.0726 ± 0.0093	**0.0527** ± 0.0024
SYM-PART simple	0.0590 ± 0.0029	0.0547 ± 0.0023	0.0534 ± 0.0023	**0.0430** ± 0.0034
SYM-PART rotated	0.0997 ± 0.2171	0.1205 ± 0.2037	0.1246 ± 0.0102	0.3081 ± 0.3649
Omni-test	0.5560 ± 0.1159	0.6623 ± 0.1386	0.3104 ± 0.0776	0.3793 ± 0.1742

Table 4: 对比算法的指标 IGDf 值

问题集编号	MMODE_CSCD mean ± std	MMOPIO mean ± std	MO_PSO_MM mean ± std	MY_ALGORITHM mean ± std
MMF1	0.0023 ± 0.0001	0.0037 ± 0.0004	0.0034 ± 0.0003	0.0025 ± 0.0001
MMF2	0.0043 ± 0.0004	0.0041 ± 0.0004	0.0092 ± 0.0012	0.0199 ± 0.0029
MMF3	0.0039 ± 0.0003	0.0037 ± 0.0009	0.0071 ± 0.0006	0.0145 ± 0.0020
MMF4	0.0023 ± 0.0001	0.0040 ± 0.0007	0.0048 ± 0.0010	**0.0023** ± 0.0001
MMF5	0.0023 ± 0.0000	0.0037 ± 0.0007	0.0033 ± 0.0003	0.0025 ± 0.0001
MMF6	0.0023 ± 0.0001	0.0034 ± 0.0003	0.0033 ± 0.0003	**0.0022** ± 0.0001
MMF7	0.0024 ± 0.0001	0.0063 ± 0.0009	0.0061 ± 0.0008	0.0025 ± 0.0002
MMF8	0.0028 ± 0.0001	0.0034 ± 0.0003	0.0031 ± 0.0001	**0.0025** ± 0.0001
MMF9	0.0104 ± 0.0006	0.0207 ± 0.0056	0.0254 ± 0.0061	0.0108 ± 0.0010
MMF10	0.0399 ± 0.0883	0.0223 ± 0.0235	0.0132 ± 0.0032	0.1049 ± 0.1410
MMF11	0.0114 ± 0.0010	0.0221 ± 0.0048	0.0198 ± 0.0034	**0.0102** ± 0.0007
MMF12	0.0021 ± 0.0001	0.0038 ± 0.0007	0.0025 ± 0.0001	**0.0020** ± 0.0001
MMF13	0.0149 ± 0.0035	0.0231 ± 0.0067	0.0177 ± 0.0041	**0.0125** ± 0.0019
MMF14	0.0914 ± 0.0029	0.1020 ± 0.0036	0.1010 ± 0.0050	**0.0848** ± 0.0017
MMF15	0.1004 ± 0.0036	0.1093 ± 0.0071	0.1060 ± 0.0047	**0.0880** ± 0.0030
MMF1_z	0.0023 ± 0.0001	0.0033 ± 0.0002	0.0032 ± 0.0001	0.0024 ± 0.0001
MMF1_e	0.0089 ± 0.0023	0.0035 ± 0.0003	0.0053 ± 0.0005	0.0050 ± 0.0009
MMF14_a	0.0894 ± 0.0021	0.0954 ± 0.0034	0.0966 ± 0.0033	**0.0851** ± 0.0018
MMF15_a	0.1028 ± 0.0048	0.1088 ± 0.0058	0.1055 ± 0.0051	**0.0893** ± 0.0030
SYM-PART simple	0.0105 ± 0.0012	0.0141 ± 0.0019	0.0172 ± 0.0024	**0.0091** ± 0.0010
SYM-PART rotated	0.0098 ± 0.0011	0.0139 ± 0.0018	0.0295 ± 0.0035	0.0156 ± 0.0022
Omni-test	0.0207 ± 0.0024	0.0113 ± 0.0014	0.0340 ± 0.0036	**0.0094** ± 0.0006

6 总结与展望

本文主要是在每个非支配层内采用分步聚类的方法进行改进，而不是简单的使用 k-means 方法进行聚类。利用距离找变量之间的相似性从而创建树，在树里来聚类，通过这样做，可以获得更准确的 CSCD 值。将该算法与 CEC 2019 多模态多目标基准集上的 3 种对等算法进行了比较，结果表明了 MY_ALGORITHM 算法在决策空间和目标空间上的优越性。未来将引入更多不同的聚类算法来对比它们的效果，希望在更多测试集上表现出优势。此外，还应研究一种新的局部 PS 的维护机制，以找到局部 PS。

References

[1] Subhodip Biswas, Souvik Kundu, and Swagatam Das. “Inducing niching behavior in differential evolution through local information sharing”. In: *IEEE Transactions on Evolutionary Computation* 19.2 (2014), pp. 246–263.

[2] Borko Bošković and Janez Brest. “Clustering and differential evolution for multimodal optimization”. In: *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2017, pp. 698–705.

[3] K. Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197.

[4] Kalyanmoy Deb and Santosh Tiwari. “Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization”. In: *European Journal of Operational Research* 185.3 (2008), pp. 1062–1087.

[5] Grant Dick. “Automatic identification of the niche radius using spatially-structured clearing methods”. In: *IEEE Congress on Evolutionary Computation*. IEEE. 2010, pp. 1–8.

[6] C. Dimopoulos. “A review of evolutionary multiobjective optimization applications in the area of production research”. In: *Congress on Evolutionary Computation*. 2004.

[7] C. Hu and H. Ishibuchi. “Incorporation of a decision space diversity maintenance mechanism into MOEA/D for multi-modal multi-objective optimization”. In: *the Genetic and Evolutionary Computation Conference Companion*. 2018.

[8] Ke et al. “An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition”. In: *IEEE Transactions on Evolutionary Computation* (2014).

[9] J. Knowles and D. Corne. “The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation”. In: *Proc Congress on Evolutionary Computation*. 1999.

[10] J. Liang et al. “A clustering-based differential evolution algorithm for solving multimodal multi-objective optimization problems”. In: *Swarm and Evolutionary Computation* 60 (2021), p. 100788.

[11] J. J. Liang et al. “Problem Definitions and Evaluation Criteria for the CEC 2019 Special Session on Multimodal Multiobjective Optimization”. In: (2019).

[12] Jing Liang et al. “A Self-organizing Multi-objective Particle Swarm Optimization Algorithm for Multimodal Multi-objective Problems”. In: *International Conference on Swarm Intelligence*. 2018.

[13] Jing J Liang, CT Yue, and Bo-Yang Qu. “Multimodal multi-objective optimization: A preliminary study”. In: *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2016, pp. 2454–2461.

[14] Qiuzhen Lin et al. “Multimodal multiobjective evolutionary optimization with dual clustering in decision and objective spaces”. In: *IEEE Transactions on Evolutionary Computation* 25.1 (2020), pp. 130–144.

[15] Y. Liu et al. “Handling Imbalance Between Convergence and Diversity in the Decision Space in Evolutionary Multimodal Multiobjective Optimization”. In: *IEEE Transactions on Evolutionary Computation* 24.3 (2020), pp. 551–565.

[16] Yiping Liu, Gary G Yen, and Dunwei Gong. “A multimodal multiobjective evolutionary algorithm using two-archive and recombination strategies”. In: *IEEE Transactions on Evolutionary Computation* 23.4 (2018), pp. 660–674.

[17] Yiping Liu et al. “A double-niched evolutionary algorithm and its behavior on polygon-based problems”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2018, pp. 262–273.

[18] Yiping Liu et al. “Handling imbalance between convergence and diversity in the decision space in evolutionary multimodal multiobjective optimization”. In: *IEEE Transactions on Evolutionary Computation* 24.3 (2019), pp. 551–565.

[19] Ole J Mengshoel and David E Goldberg. “The crowding approach to niching in genetic algorithms”. In: *Evolutionary computation* 16.3 (2008), pp. 315–354.

[20] M. Pal and S. Bandyopadhyay. “Decomposition in Decision and Objective Space for Multi-Modal Multi-Objective Optimization”. In: *Swarm and Evolutionary Computation* 62.1 (2021), p. 100842.

[21] G. Rudolph. “Evolutionary Search under Partially Ordered Fitness Sets”. In: *IN PROCEEDINGS OF THE INTERNATIONAL SYMPOSIUM ON INFORMATIONENCE INNOVATIONS IN ENGINEERING OF NATURAL AND ARTIFICIAL INTELLIGENT SYSTEMS* (2001).

[22] Bruno Sareni and Laurent Krahenbuhl. “Fitness sharing and niching methods revisited”. In: *IEEE transactions on Evolutionary Computation* 2.3 (1998), pp. 97–106.

[23] Y. Tan, Y. Shi, and Q. Tang. “A Self-organizing Multi-objective Particle Swarm Optimization Algorithm for Multimodal Multi-objective Problems”. In: 10.1007/978-3-319-93815-8.Chapter 52 (2018), pp. 550–560.

[24] R. Tanabe and H. Ishibuchi. *A Framework to Handle Multimodal Multiobjective Optimization in Decomposition Based Evolutionary Algorithms*. 2020.

[25] R. Tanabe and H. Ishibuchi. “A niching indicator-based multi-modal many-objective optimizer”. In: *Swarm and Evolutionary Computation* 49 (2019).

[26] Ryoji Tanabe and Hisao Ishibuchi. “A Decomposition-Based Evolutionary Algorithm for Multi-modal Multi-objective Optimization: 15th International Conference, Coimbra, Portugal, September 8‑12, 2018, Proceedings, Part I”. In: 2018.

[27] Yi HUJie WANGJing LIANGKunjie YUHui SONGQianqian GUOCaitong YUEYanli WANG. “A self-organizing multimodal multi-objective pigeon-inspired optimization algorithm”. In: 中国科学 000.007 (2019), P.73–89.

[28] Caitong Yue, Boyang Qu, and Jing Liang. “A multiobjective particle swarm optimizer using ring topology for solving multimodal multiobjective problems”. In: *IEEE Transactions on Evolutionary Computation* 22.5 (2017), pp. 805–817.

[29] Q. Zhang and L. Hui. “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition”. In: *IEEE Transactions on Evolutionary Computation* 11.6 (2008), pp. 712–731.

[30] Eckart Zitzler and Lothar Thiele. “Multiobjective optimization using evolutionary algorithms —A comparative case study”. In: *Springer, Berlin, Heidelberg* (1998).