分 类 号： TP391
学校代码： 10109
密　　级： 公 开

太原科技大学硕士学位论文
（学术型）

学位论文题目： 高维多目标拐点鸽群算法

英文题目： **Research on many-objective knee point pigeon-inspired optimization algorithm**

研 究 生 姓 名 ： 赵丽红

导师姓名及职称： 王丽芳 副教授　崔志华 教授

培　养　单　位 ： 计算机科学与技术学院

学　科　专　业 ： 计算机科学与技术

论文提交日期： **2022** 年 **4** 月

论文答辩日期： **2022** 年 **5** 月 **23** 日

答辩委员会主席： 曾建潮

中 文 摘 要

鸽群算法作为近几年提出的一种新型群智能优化算法。原理简单、参数较少等特性使其在多目标优化问题中表现良好，受到了学术界和工业界的广泛认可。近年来实际工程问题变得越来越复杂，在解决非决策者偏好的高维多目标优化问题时，随着目标空间维度的增加，现有的高维多目标鸽群算法存在求解性能不足、多样性和收敛性难以平衡以及选择压力欠缺等问题。由此做了以下研究工作：

1. 现有高维多目标鸽群算法解决非决策者偏好的高维多目标优化问题时，在传统的 Pareto 机制上增加了其他策略来提高算法的选解能力，但随着目标个数的增加，种群中的解几乎互相都是非支配的，因此提高算法选择压力与收敛能力成为了亟待解决的问题。针对上述问题，考虑到拐点作为较大超体积值的偏置，它的性能天生优于种群中的其他个体，本文提出了一种基于拐点支配的高维多目标鸽群算法。通过引入基于拐点支配的环境选择策略，有效提高了算法的选择压力；通过结合不同分布特性的速度位置更新策略，动态调整算法在目标空间中的搜索方向，从而提高了算法的整体搜索性能。实验分析表明本文所提算法在个体选择和逼近真实 Pareto 前沿方面均有较好的性能。

2. 针对当目标个数急剧增加时，现有的高维多目标鸽群算法在多样性和收敛性能方面难以实现平衡，以及进一步提高算法在处理高维多目标优化问题时的求解性能。本文提出了基于 KIGD 指标的竞争高维多目标鸽群算法。该算法通过基于 KIGD 指标的环境选择策略来实现对拐点及其周围区域解的识别从而提高解集覆盖拐点区域的多样性；通过引入基于竞争思想的种群更新策略，从而提高算法收敛性能。所进行的对比实验的结果也证明了所提的基于 KIGD 指标的竞争高维多目标鸽群优化算法具有的优越性能。

3. 为进一步解决高维多目标鸽群算法中目标数量的增加导致算法收敛性和多样性冲突的问题，本文在上述方法的基础上提出了一种基于多选择策略的高维多目标鸽群优化算法。从并发集成角度设计了基于多选择策略的精英保留策略，通过集成多种优秀选择策略，可以针对不同问题的特性，有目的的选用不同的算子来提高求解效率，同时能一定程度的保证种群的收敛性和多样性。此外，也采用外部归档集存储最优非支配个体。实验分析表明本文所提的基于多选择策略的高维多目标鸽群优化算法具有相对较好的综合性能。

关键词：高维多目标鸽群算法；高维多目标优化问题；环境选择；选择压力；收敛性；多样性

ABSTRACT

Pigeon-inspired optimization algorithm as a swarm intelligence algorithm proposed in recent years. Because of its good performance in multi-objective optimization problems, it has been widely recognized by academia and industry. In recent years, practical engineering problems have become more and more complex. When solving many-objective optimization problems that are not favored by decision-makers, with the increase of the number of objectives, the algorithm has problems of insufficient performance, intensified conflict between diversity and convergence, and lack of selection pressure. Aiming at the above problems, the following research work has been done to address the above issues:

1. When the existing many-objective pigeon-inspired algorithm solves the many-objective optimization problems that are not preferred by decision-makers, although other strategies are added to the traditional Pareto mechanism to improve the solution selection ability of the algorithm, with the increase of the number of objectives. The solutions in the population are almost non-dominated to each other, so improving the algorithm selection pressure and convergence ability has become an urgent problem to be solved. Aiming at the above problems, considering that the knee point is a bias for a larger hypervolume, its performance is inherently better than other individuals in the population. This paper proposes a knee point-driven many-objective pigeon-inspired algorithm. By introducing the environment selection strategy based on knee-oriented dominance, the selection pressure of the algorithm is effectively improved; by combining the velocity and position update strategies with different distributions, the population evolution direction is dynamically adjusted in different stages of iteration, thereby improving the overall search performance of the algorithm. Experimental results show that the proposed algorithm has good performance in individual selection and approximation to the true Pareto Front.

2. In view of the deficiency of the existing many-objective pigeon -inspired algorithm in balancing convergence and diversity when the number of objectives

increases sharply, and to further improve the solution performance of the algorithm when dealing with many-objective optimization problems. This paper proposes a KIGD indicator based many-objective pigeon-inspired algorithm. The algorithm realizes the identification of the solution of the knee point and its surrounding area through the environmental selection strategy based on KIGD indicator, thereby improving the diversity of the solution set covering the knee point region. A population renewal strategy based on competition is introduced to improve the selection pressure of the algorithm and ensure its convergence performance. The simulation results also prove the superior performance of the proposed algorithm based on KIGD indicator.

3. In order to further solve the problem of algorithm convergence and diversity conflict caused by the increase of the number of objectives in the many-objective pigeon-inspired algorithm, this paper proposes a many-objective pigeon-inspired based on multiple selection strategies. From the perspective of concurrent integration, an elite retention strategy based on multiple selection strategies is designed. By integrating a variety of excellent selection strategies, different operators can be selected according to the characteristics of different problems to improve the solution efficiency, and at the same time, convergence and diversity of population can be guaranteed to a certain extent. In addition, an external archive set is also used to store the elite non-dominated individuals. The experimental analysis shows that the many-objective pigeon-inspired optimization algorithm based on multiple selection strategy proposed in this paper has relatively good comprehensive performance.

Keywords: Many-objective pigeon-inspired algorithm; Many-objective optimization problem; Environmental selection; Selection pressure; Convergence; Diversity

目　录

第一章 引 言

1.1 高维多目标优化问题

随着社会以及工业的发展，传统的工业问题变得愈加复杂，众多专家学者通过数学建模来对实际工业问题[1, 2]进行相应的求解，从而得到解决问题更好的方案。而在建模过程中，研究者们开始更加关注具有冲突性的问题与需求，提出了多目标优化问题[3, 4]（Multi-objective optimization problems, MOPs）的概念。随之，相关求解 MOPs 的策略与方法被广泛的提出与应用。基于多个问题的冲突性以及非关联性，就导致了当问题的一个目标取得最优值时，往往在其他几个目标上性能相对较差，如何得到一组相对性能较佳的解决方案成为了研究的重点以及热点。因此，研究者们创造性的提出了一些基于不同机制的多目标进化算法[5]（Muiti-objective optimization algorithm, MOEAs）。此外，后续针对算法性能改进以及将其适用到实际工业应用也成为了学者们进一步研究的课题。但由于多目标优化问题仅考虑目标个数为 2-3 个的情况，使其在近几年的研究中愈发存在不足，现在的实际工程问题因为计算机的大量普及以及互联网的发展，除了要考虑问题本身的需求，机器设备与有限资源也要加以分析，使得问题的规模更大，模型更加复杂[6]。因此，对高维多目标优化问题[7]（Many-objective optimization problems, MaOPs）进一步研究，并提出了许多求解 MaOPs 的相关算法[8, 9]。本文的研究也是在此基础上进行的，为了更好的阐述后续的研究内容，本节首先对相关的一些定义进行简单的介绍。相关定义如下：

定义 1.1：多目标优化问题 MOPs 定义为

$$\min F(x)=\min\left(f_1(x), f_2(x),\cdots, f_M(x)\right)^T \tag{1.1}$$

$$\begin{cases} g_i(X)\geq 0, i=1,2,\ldots,k \\ h_j(X)=0, j=1,2,\ldots,p \end{cases} \tag{1.2}$$

$$subject\ to\quad X=(x_1,x_2,\cdots,x_n)\in\Omega\subseteq R^n \tag{1.3}$$

其中，$X=(x_1,x_2,\cdots,x_n)$是n维决策空间Ω中的解向量。$X_i=[x_{i1},x_{i2},\cdots,x_{iD}]$代表在第$M$个目标函数上的目标函数值。公式 1.2 为问题需要满足的相关约束条件。MaOPs 是在上述概念下将目标个数扩展到四个及以上的情况。

定义 1.2（*Pareto* 支配）：对于种群中的解向量$X_i=[x_{i1},x_{i2},\cdots,x_{iD}]$和$X_i=[x_{i1},x_{i2},\cdots,x_{iD}]$，当满足下列条件，则称$X^a$ *Pareto* 支配X^b，表示为 $X^a\prec X^b$。

$$\forall i\in\{1,2,\ldots,m\}, f_i(x)\leq f_i(y)\wedge j\in\{1,2,\ldots,m\}, f_i(x)<f_i(y) \tag{1.4}$$

定义 1.3（*Pareto* 最优解）：当且仅当满足以下条件，决策变量 $x^* \in \Omega$ 被认为是 Pareto 最优解。

$$\nexists x \in \Omega : x \prec x^* \tag{1.5}$$

定义 1.4（*Pareto* 前沿面）：MOPs 中，Pareto 前沿面 PF 的定义如下：

$$X_i = \left[x_{i1}, x_{i2}, \cdots, x_{iD}\right] \tag{1.6}$$

1.2 标准鸽群算法

鸽群算法[10]（Pigeon-inspired optimization algorithm, PIO）是一种较为新兴的群智能算法， 在 2014 年由段海滨教授提出并用于解决无人机路径规划问题。通过模拟鸽子自主归巢行为，进一步将鸽子利用地磁场以及太阳高度等信息实现远距离寻的行为进行建模。由于其原理简单、参数少、易实现以及比其他群智能算法收敛速度更快等特点，被广泛应用在无人机编队、参数控制以及图形处理等多个领域中。

鸽群优化算法仿真模拟了鸽子在寻找目的地时利用不同导航工具在飞行的不同阶段，从而最终寻优的行为。基于鸽子在寻的过程中的飞行行为，鸽群算法的数学模型主要包括两个独立工作的操作算子模型：

1）地图和指南针算子（map and compass operator）

地图和指南针操作算子模拟了鸽群在距离目的地较远时，利用地磁场和太阳高度调整飞行方向，从而实现寻找目的地的行为。在远离目的地时，鸽群首先利用磁性物体感受磁场，进而绘制飞行地图；然后通过判断飞行过程中太阳的高度来进一步调整飞行方向。而随着鸽群离目的地越来越近，太阳和磁场对鸽群个体的影响越来越少。

2）地标算子（landmark operator）

地标操作算子模拟了鸽群在距离目的地较近时，利用附近地标来调整飞行方向。鸽群中对地标附近信息熟悉的鸽子，可以径直飞向目的地；否则，鸽群将跟随熟悉地标信息的鸽子飞行，最终到达目的地。

在鸽群优化算法的模型中，利用虚拟的鸽子个体模拟利用上述两种操作算子的导航飞行过程。首先进行地图和指南针操作算子的运行过程，初始化种群中鸽子的位置 $X_i = \left[x_{i1}, x_{i2}, \cdots, x_{iD}\right]$ 和速度 $V_i = \left[v_{i1}, v_{i2}, \cdots, v_{iD}\right]$，其中，$i = 1, 2, \ldots, N$，即种群大小为 N，决策空间维度为 D。在每一次迭代过程中，更新鸽子的位置和速度。

鸽群中每个个体的速度位置更新如下所示：

$$V_i(t) = V_i(t-1) \cdot e^{-Rt} + rand\left(X_{gbest} - X_i(t-1)\right) \tag{1.7}$$

$$X_i(t) = X_i(t-1) + V_i(t) \tag{1.8}$$

其中，R 代表地图和指南针算子，且其取值介于 $[0,1]$ 之间，$rand$ 是介于 $[0,1]$ 之间的随机数，t 是当前种群迭代次数。X_{gbest} 代表前 $t-1$ 代中，利用所有鸽子的位置信息得到的全局最优位置。地图和指南针操作算子模型一直迭代运行直到达到最大迭代次数终止运行后算法进入地标操作算子模型。

地标操作算子模型模拟了鸽群飞行后期逐渐靠近目的地的寻的行为，利用上述地图和指南针算子模型得到的种群和其中的速度位置信息进行下一步的种群更新。在地标操作算子中，每一次迭代过程中鸽群的数量 N 都会减少一半，通过适应度值舍弃性能不好即远离目的地的鸽子。对种群中的所有个体计算其位置中心 X_{center}，将其作为引导种群前进的参考方向。

$$N(t)=\frac{N(t-1)}{2} \tag{1.9}$$

$$X_{center}(t)=\frac{\sum_{i=1}^{N(t)} X_i(t)\,Fitness\left(X_i(t)\right)}{N(t)\sum_{i=1}^{N(t)} Fitness\left(X_i(t)\right)} \tag{1.10}$$

$$X_i(t)=X_i(t-1)+rand\left(X_{center}(t-1)-X_i(t-1)\right) \tag{1.11}$$

其中，$Fitness\left(X_i(t)\right)$ 代表当前迭代中解的适应度值，从而进行解的比较。$rand$ 为介于 $[0,1]$ 之间的随机数，t 是当前种群迭代次数。$N(t)$ 为第 t 次迭代中的种群个数。当循环达到最大迭代次数后，地标操作算子模型运行终止，根据当前种群中的个体适应度比较得到性能最好的个体。

从上述鸽群算法数学模型可以看出，鸽群算法具有参数较少的特点。因此在解决复杂问题时，无需配置太多参数。鸽群算法中地图和指南针操作算子在迭代运行中能够对全局进行搜索，利用个体自身的速度和全局最优位置尽可能对目标空间进行全局搜索；地标操作算子在迭代后期更倾向于局部收敛，一方面利用种群减半策略提高种群收敛速度，一方面利用种群的位置中心位置更新种群中的个体，从而提高其局部搜索能力。

鸽群算法的流程图如图 1.1 所示，具体步骤如下：

第一步 执行种群初始化操作，初始化相关参数，产生 n 个个体 $X_i=[x_{i1},x_{i2},\cdots,x_{iD}]$；

第二步 个体的速度和位置信息更新，计算个体的适应度值，并更新种群中全局最优 x_{gbest} 和个体历史最优 X_{gbest}；

第三步 判断是否达到地图和指南针操作算子的最大迭代次数，若达到，进入下一步；否则，返回第二步；

第四步 更新种群中个体总数，计算种群中个体位置中心 X_{center}；

第五步 更新种群中个体的速度位置信息，以及种群中全局最优 X_{gbest} 和个体历史最优 X_{pbest}；

第六步 判断是否达到地标操作算子的最大迭代次数，若达到，则终止算法返回最好适应度值对应的结果；否则，返回第四步；

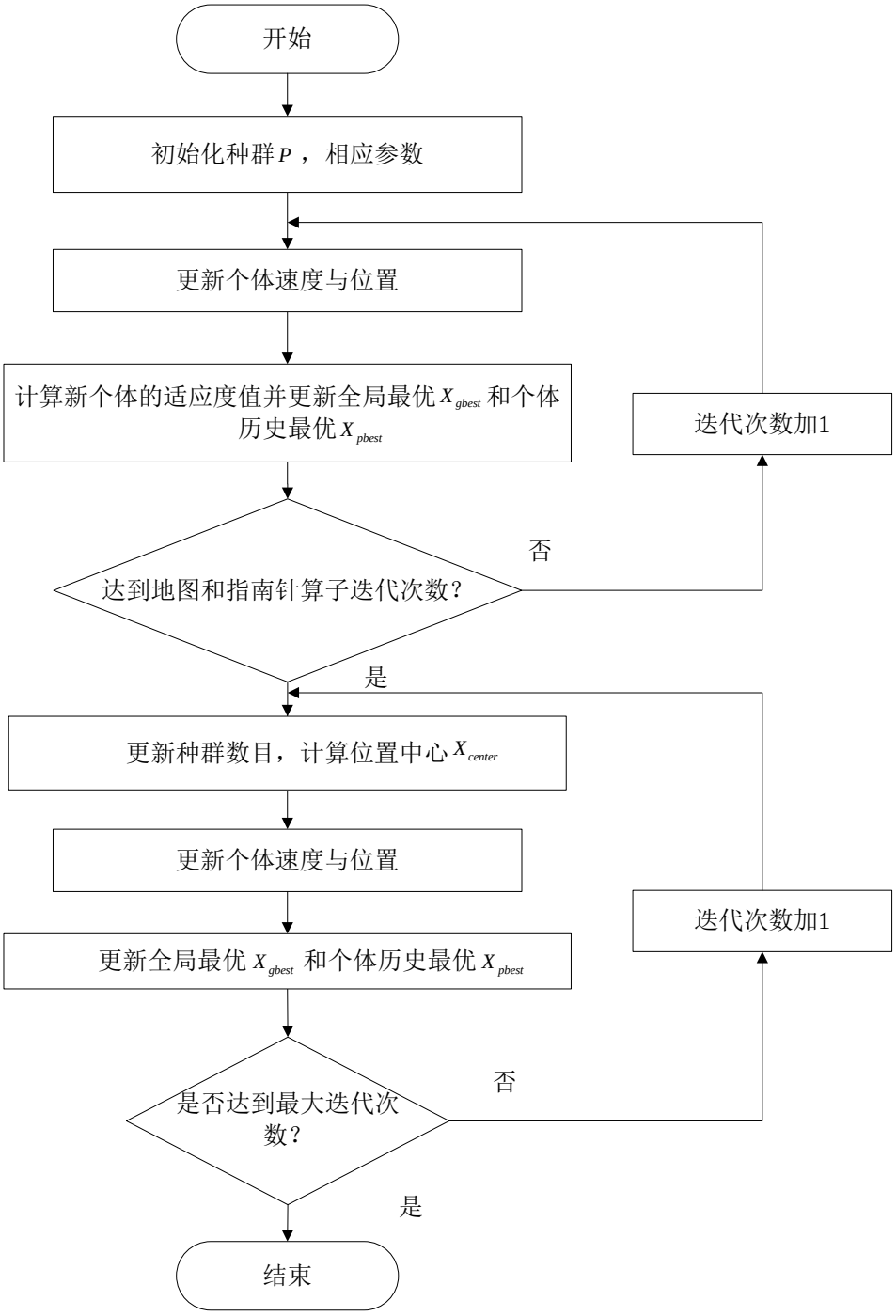


图 1.1 标准鸽群算法流程图

Fig 1.1 Flow chart of standard pigeon-inspired algorithm

1.3 国内外研究现状

自从鸽群算法[10]提出以来,由于其参数简单且算法鲁棒性较强，以及算法在许多单目标工业问题中的良好性能，专家学者对其进行了进一步的研究，包括相关算法改进和现实应用，尤其在无人机编队、图像处理、参数优化等领域。其中，Zheng 等人[11]于 2016 年将 PIO 算法应用于解决蛋白质的二级结构预测问题。白晨等人[12]利用 PIO 算法解决了含分布式电源的配电网的状态估计问题。胡春鹤等人[13]利用 PIO 算法解决图像分割问题，通过对图像分割建模，优化分割阈值，从而求解最优参数，从而得到图像分割的最佳阈值。唐悦等人[14]将 PIO 算法应用到对空天飞行器的飞行品质评估问题中，对所构建的动力学模型进行等效拟配求解。

由于标准 PIO 算法的收敛速度较快，在复杂的搜索空间中易陷入局部最优。Li 等人[15]在 2014 年提出了一种基于边缘保留滤波函数 EPF 的改进模拟退火鸽群算法(SAPIO)来解决无人机目标检测问题，通过引入与鸽群数量相关的概率 来判断迭代处于何种操作算子模型，通过产生的随机数和 进行比较，动态决定执行的操作算子类型；进而引入高斯扰动和模拟退火机制来改进标准 PIO 算法易于陷入局部最优的问题。Zhang 等人[16]在 2016 年提出了一种基于 Levy 飞行的鸽群算法来解决小型无人直升机中的自抗扰控制器的参数优化问题，将 Levy 飞行的搜索特性引入到指南针操作算子中，并使用 Levy 飞行将整个搜索空间进行扩大，同时引入 Logsig 函数，提高收敛的同时解决标准 PIO 算法易陷入局部最优的问题。杨之元等人[17]利用基于莱维飞行的鸽群算法解决无人机编队中比例-积分-微分仿雁群编队控制器的参数优化问题。段海滨等人[18]于 2015 年提出了基于捕食逃逸鸽群算法用于解决无人机紧密编队控制问题，通过引入了捕食逃逸机制改善标准 PIO 算法易陷入局部最优的问题。Zhang 等人[19]提出了一种改进的基于被捕食者-捕食机制的鸽群算法(PPPIO)来解决无人机三维路径规划问题，利用了捕食者-被捕食者的概念，使子代个体广泛分布在目标空间中，既避免了个体的早熟，又提高了寻优速度。

Yang 等人[20]于 2018 年提出了一种基于柯西突变的鸽群算法(CMPIO)来解决舰载机自动着陆系统多层参数设计问题，通过引入一个固定的柯西分布参数决定鸽子位置的随机改变能力，利用不同的柯西突变机制对 PIO 中的两个算子进行改进，进而减少算法陷入局部收敛的概率。同年，段海滨等人[21]将上述基于柯西变异的鸽群算法(CMPIO)应用于解决大型民用飞机的滚动时域控制中。胡明月等人[22]利用自适应参数和柯西扰动解决了低压配电台区三相负荷不平衡问题。Sun 等人[23]于 2020 年提出了一种拟仿射变换鸽群算法(QT-PIO)，利用仿射算法 QUATR 使个体在优化过程中以矩阵的形式更新自己的位置；在地标算子模型中采用两种更新策略，对标准 PIO 算法中将被舍弃的个体进行类

似 QUATRE 算法的学习策略，有效地避免了 PIO 算法陷入局部最优的问题。Tian 等人[24]提出了一种紧凑鸽群算法(CPIO)来解决在梯级水电站短期发电优化中减少内存使用和参数选择的问题，通过构建总体解决方案的概率来复制基于群体算法的操作，优化过程将实际种群的概率表示编码为虚拟对应物，即利用一个紧凑 PIO 的概率模型来表示鸽子种群的所有解集，从而不仅提高了时间效率，而且减少了硬件内存。Hai 等人[25]提出了一种基于进化博弈论的自适应鸽群算法(EGTPIO)，利用博弈特性，开发了鸽子个体之间的双重战略进化博弈，将 PIO 算法中的两个操作算子作为博弈策略，通过对一个特定的算子进行平均成本值的组合构成收益矩阵，利用收益矩阵动态调整 PIO 中操作算子的选择，从而加快算法收敛速度。

而随着现实工业问题的复杂性日益提升，对应的问题模型也日趋复杂，研究者们提出了相应的多目标鸽群算法[26, 27, 28]。其中，Qiu 等人[29]在 2015 年提出了多目标鸽群算法(MPIO)用于解决无刷直流电机中的多目标参数优化问题，也采用了 Pareto 支配机制和拥挤度操作算子来对鸽子个体进行选择，同时提出了合并算子的定义，将单目标鸽群算法中的地图和指南针操作算子以及地标操作算子进行合并，利用过渡因子使其能够实现在两个操作算子运行过程之间的平稳过渡。Fu 等人[30]提出了一种多目标鸽群算法解决了模糊生产调度问题，采用非支配解作为领导鸽子个体指定的候选解，并利用特殊的拥挤距离确保解在目标空间和相应的决策空间均具有良好的分布。此外，利用基于索引的环形拓扑提高算法的收敛速度。Duan 等人[31]提出了一种基于极限环的多目标鸽群算法(CMMOPIO)，设计了基于极限环的机构和突变体机制，加强进化过程中的探索能力，同时利用双知识库来对非支配解进行存储和选择，从而指导鸽子的飞行。由于基于极限环的突变机制，既提高收敛速度和精度，又提高了算法的种群多样性。Qiu 等人[32]提出了一种改进的多目标鸽群算法解决无人机群在障碍物之间分布问题，利用分层学习思想，通过 Pareto 排序对种群中的个体划分层级，每个鸽子个体赋予相应的层级，高阶层鸽子个体将带领低阶层鸽子个体飞行，从而提高算法进化效率。

Shang 等人[33]在提出了一种以负比关联(Negative Ratio Association, NRA)和比值切割准则(Ratio Cut, RC)作为目标函数的多目标鸽群算法(MOPIO)来解决社区检测问题。在 MOPIO 中，利用遗传算子重新定义鸽子的表示和更新。在每次迭代中，计算每只鸽子的 NRA 和 RC，并利用 Pareto 排序方案判断非支配解，以供后期交叉。设计了一种基于全局最佳和个人最佳的交叉策略，提出了一种补偿系数，以稳定完成两个操作算子的工作过渡。当满足终止条件时，采用超前选择策略从最优解集中确定最终结果。Tong 等人[34]提出了一种基于鸽群优化和差分进化的改进多目标鸽群算法用于解决无人机路径规

划和自主编队问题，将无人机路径规划数学模型设计为多目标优化模型，引入路径长度、路径弯度和路径风险三个指标；在此基础上，利用差分进化的变异策略和改进多目标鸽群算法对可行路径进行优化，并利用 Pareto 支配选择鸽子的全局最优位置。

而随着工业大数据的提出以及物联网+的提出，现实工业问题变得更加复杂，针对解决 MaOPs 时，现有的多目标鸽群算法求解性能不佳，无法得到收敛性能更好的个体。Cui 等人[35]提出了一种高维多目标鸽群算法(MAPIO)，将鸽群算法扩展到了解决高维多目标问题上，采用平衡适应度估计 BFE 策略以及模二进制交叉和多项式变异等策略提高算法求解高维 MaOPs 的性能。现有的高维多目标鸽群算法在一定程度上弥补了在该领域的空缺，但当目标个数急剧增加时，传统的 Pareto 机制对个体的选择压力以及算法收敛性和多样性平衡方面均存在一定问题，基于以上原因，本文针对上述问题，论文从提高算法选择压力以及逼近真实 Pareto 前沿方面的能力、提高算法收敛性和多样性等综合性能以及提高算法解决不同类型问题的性能等方面分别对现有高维多目标鸽群算法进行相应的改进。

1.4 文章主要工作以及内容安排

标准鸽群算法作为最近几年提出的新型群智能优化算法，因其参数较少，原理简单，算法鲁棒性较强等特性被众多专家学者所研究。而随着现实工业问题变得越来越复杂，所建模型的目标空间维度越来越高，高维多目标优化问题成为了大家研究的热点。针对现有高维多目标鸽群算法存在的问题，本文尝试对其进行相应改进，主要工作如下安排：

第一章：主要介绍了高维多目标优化问题的相关概念以及定义；然后介绍了标准的鸽群算法及其数学模型；最后，介绍并分析了鸽群算法现有的国内外研究现状。

第二章：针对现有高维多目标鸽群算法在求解 MaOPs 时，由于现有 Pareto 支配机制选择压力不足导致的个体选择困难以及无法逼近真实 Pareto 前沿的问题，本文提出了基于拐点支配的高维多目标鸽群算法，并通过仿真实验进一步验证了所提方法在提高算法选择压力以及逼近真实 Pareto 前沿方面的能力。

第三章：针对现有高维多目标鸽群算法在解决 MaOPs 时，由于种群迭代后期非支配解大量存在导致的收敛性和多样性冲突的冲突问题，考虑对局部感兴趣点及周围区域进行选择，提出了基于 KIGD 指标的竞争高维多目标鸽群算法，并在标准测试集验证了所提算法提高算法收敛性和多样性等综合性能方面的性能。

第四章：在前两章策略提高现有高维多目标鸽群算法性能的基础上，针对求解具有不同特征测试示例时性能不足的问题，提出了基于多选择策略的高维多目标鸽群算法，

并跟其他算法在标准测试集上进行比较，进一步衡量了本章提出的算法在求解不同特性问题时的收敛性和多样性等综合性能。

第五章：总结与展望。总结本文现有研究工作，并提出了未来研究方向和目标。

论文结构安排如图 1.2 所示。

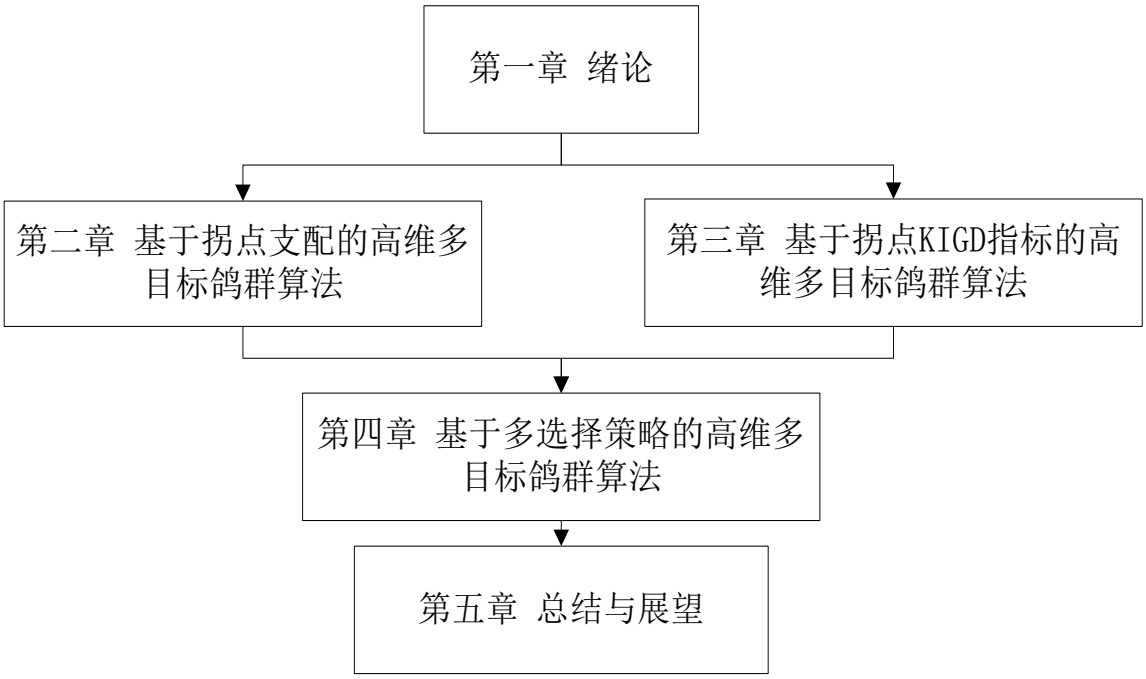


图 1.2 论文结构安排

Figure 1.2 The structure of this paper

第二章 基于拐点支配的高维多目标鸽群算法

本章首先通过研究非支配解在目标种群中的占比问题，进一步分析了现有高维多目标鸽群算法在目标个数增加的情况下存在的选择压力不足的问题。然后，结合拐点机制的特点[36]，从环境选择的角度设计了一种基于拐点支配的环境选择策略，并结合不同分布的特性设计了一种新的速度位置更新公式。最后，将所提算法与其他算法在两类对比实验中进行比较，分析了其在面向拐点的测试集 PMOPs[37]与经典测试集 DTLZ[38]和 WFG[39]上的性能。

2.1 问题分析

近年来，非决策者偏好的 MaOPs[40]在现实应用中变得越来越常见，决策者偏好不明确的情况下，无法采用现有的降维方式对目标个数进行约减，传统的多目标优化算法已经不再适应于求解上述问题。现有高维多目标鸽群算法的提出在一定程度上为解决现实 MaOPs 提供了解决方案，但是基于 Pareto 支配关系的个体选择策略也使得算法本身的性能并不能得到有效的保证。虽然已经采取了一些措施来改善 Pareto 支配机制的性能，但个体的选择和收敛并没有得到保证，并且产生的非支配解的局限性在 MaOPs 中更加突出。图 2.1 展示了随着目标个数增加，现有高维多目标鸽群算法在不同测试问题上非支配解占种群总个数的比例。

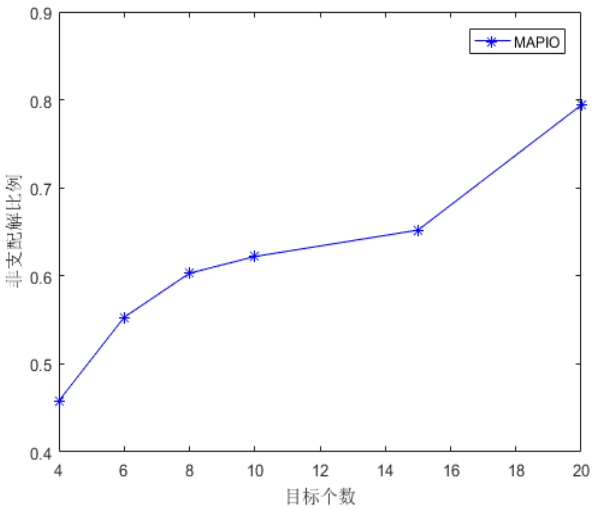


图 2.1 高维多目标鸽群算法随着目标个数增加在不同测试问题上的非支配解占比

Fig 2.1 The non-dominated solution ratio of many-objective pigeon-inspired algorithm on different test problems with increasing number of objectives

从图 2.1 可以看出，随着目标个数的增加，非支配解在种群中的占比逐渐增加，这也导致在当前种群中对个体进行进一步的选择变得愈加艰难，而随着迭代次数的增加，

现有的 Pareto 支配机制已经无法提供足够的选择压力。而这种情况下所产生的抗支配解[41]（极端条件下在一个目标上接近最优但在其他至少一个目标上性能较差），会导致现有的基于 Pareto 支配的算法在目标空间中难以找到趋向于真实 Pareto 前沿的个体。

表 2.1 常见的支配策略研究

Table 2.1 Research on common dominance strategies

类型	支配策略	原理
松弛 Pareto 支配方式	模糊 Pareto 支配[42]	提出了模糊帕累托支配关系，利用一种泛型排序方法以非标度、非对称、集相关方式分配支配程度给解集中的向量。
改进传统 Pareto 支配方式	θ 支配[43]	利用参考点机制的新支配方式，种群中的个体归属于不同的参考点，对同属于一个参考点的簇内个体进行 θ 支配排序。优先选择各簇中适应度值较高的解，保证所选解在这些簇中尽可能均匀分布。
	γ 支配[44]	利用参考点策略，在保持 Pareto 支配排序的同时，γ 支配能够根据决策者的偏好将搜索引导到 Pareto 前沿最优的部分，选择更接近参考点的解决方案。通过改变 ASF 的权重向量，可以实现对某些目标的偏差。
	grid 支配[9]	目标空间网格化，并引入了三种基于网格的选择准则：grid 支配、网格排序、网格坐标点距，比较个体在支配和环境选择过程中的优势。
部分区域个体偏好的支配方式	α 支配[41]	分别对每个子区域个体采用 α 支配排序，寻找潜在的拐点区域，去除抗支配解，从而引导搜索到多个潜在的拐点区域。

一旦基于 Pareto 支配的选择准则不能有效区分解决方案，MOEAs 往往依赖于次要准则，从而得到一组多样性较好的解决方案[8, 45]，但是这组解并非是 Pareto 最优解集。因此，一些提高算法选择压力和整体收敛性的支配策略被研究者们提出,如表 2.1 所示。其中，模糊 Pareto 支配是在传统 Pareto 支配关系上通过引入个体的支配程度来对种群进行整体排序，从而提高选择压力。θ 支配与 γ 支配均是基于参考点策略提出的新的支配方式，前者通过将个体附着到距离最近的参考点，从而将种群中的个体分为不同的簇，并利用 θ 支配对簇内个体进行进一步的选择。后者则是考虑决策者的偏好，使个体朝着

更接近于参考向量的方向进化，并通过改变 ASF 的权重向量实现对某些目标的偏差。grid 支配是在对目标空间网格化之后，利用包括网格支配在内的三个选解准则来提高算法的个体选择压力。α 支配则是从对部分优势区域个体进行选择的角度所提出的新的支配策略。通过是利用拐点机制，实现消除抗支配解以及边界解从而提高算法选择压力。

上述机制都在一定情况下提高了相应算法的选择压力，证明了一个有效的环境选择策略能够在算法后期难以收敛的情况下，提高算法的选解能力与收敛能力。但随着高维多目标优化问题中目标个数急剧增加，对整个种群中的个体比较存在很大的困难，因此考虑对种群中的局部感兴趣区域进行进一步分析，极值点、边界点、拐点以及拐点周围区域均是我们分析的重点。而拐点区域作为 Pareto 前沿的真子集，它能够有效降低所要比较的个体规模。因此，将对整个种群的比较转为对局部区域个体的分析与比较从而提供足够的选择压力以及提高算法逼近 Pareto 真实前沿的能力成为需要解决的两个关键问题。为此，本章在 Pareto 支配的基础上设计了一个基于拐点支配的高维多目标鸽群算法，通过采用基于拐点支配的环境选择策略来提高算法的整体选择压力，同时利用不同分布的特性来提高算法在整个迭代过程中的搜索性能，使其能够更近一步逼近 Pareto 真实前沿。

2.2 基于拐点支配的高维多目标鸽群算法

在本节中，详细描述所提出的基于拐点支配的高维多目标鸽群算法（KnMAPIO）。首先，在 2.2.1 节中分析了拐点以及拐点支配机制。然后，在 2.2.2 节中描述 KnMAPIO 的总体框架。主要包括以下两个个方面：基于拐点支配的环境选择策略来提高个体的选择压力；新的速度位置更新公式来提高算法在目标空间中的搜索性能。分别在论文中的 2.2.3 和 2.2.4 节进行详细描述。最后，该算法的计算复杂度在 2.2.5 节中进行了相应讨论。

2.2.1 基于拐点机制的研究分析

在非决策者偏好问题中，现有的 Pareto 支配在目标个数增加时，选择压力急剧下降，为提高现有高维多目标算法的求解性能和选择压力，提出了大量的 MOEAs 去寻找 Pareto 最优前沿[46, 47, 48]中的局部区域或感兴趣的点。在各种偏好中，拐点通常被认为是 Pareto 最优前沿的感兴趣点，大量的研究工作致力于寻找拐点或拐点区域(拐点的邻近区域)。Pareto 最优前沿的拐点指的是具有最大边际收益率的解决方案，这意味着这种解决方案在一个目标上的微小改进，会导致在其他至少一个目标上的严重退化。拐点作为较大超体积值的偏置，它的性能在一定情况下要优于其他个体，被优先选择的几率最大。因此，将拐点及其周围点作为个体选择的首要标准，利用个体与极值点形成的超平面之间的距离来衡量个体的性能，并将其转化为角度信息，进而扩展到拐点支配。相关定义如下：

定义 2.1（拐点 *knee points*）：拐点 k 定义为从个体极小的凸包（CHIM）到由极值点所形成的超平面 H 距离最远的点[49]。

$$k = \arg\max_{p}\left(d\left(K,H\right)\right) \tag{2.1}$$

其中 K 为种群中的个体，$d\left(K,H\right)$ 为种群中的个体到极值点所形成的超平面 H 的欧式距离。图 2.1 显示了 Pareto 前沿的拐点，其中 A 和 C 是单个目标上的极值点，构成了了超平面 H 。d 是 Pareto 前沿面中的拐点 K 到超平面 H 的最大距离。

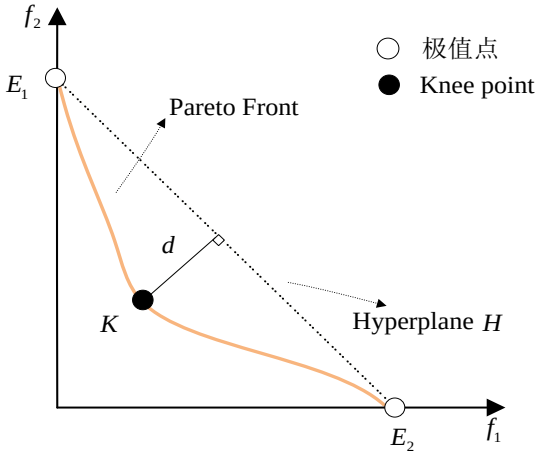


图 2.2 拐点示例图

Fig. 2.2 Example of knee point

定义 2.2（拐点支配）[49]：对于位于某个极小值的凸区域内的个体 M 和 N，如果个体 M 拐点支配个体 N，则一定满足以下条件：

$$\begin{cases} \mu\left(M,N\right)=\left\langle \overrightarrow{N_{id}M},\overrightarrow{MN}\right\rangle-\tau\left(\max\limits_{i=1.....m}\left\{\delta_i\left(M\right)\right\}+\min\limits_{i=1.....m}\left\{\delta_i\left(M\right)\right\}\right)q \\ \delta_i\left(M\right)=\arctan\left(\dfrac{\sqrt{\sum_{j=1,j\neq i}^{m}\left(f_j\left(M\right)-f_j\left(N_{id}\right)\right)^2}}{\left|f_i\left(M\right)-\max f_i\left(E\right)-\varepsilon\right|}\right) \end{cases} \tag{2.2}$$

其中 $\mu\left(M,N\right)<0$ 代表着个体 M 拐点支配个体 N。$\left\langle \overrightarrow{N_{id}M},\overrightarrow{MN}\right\rangle$ 表示两个向量 $\overrightarrow{N_{id}M}$ 和 $\overrightarrow{MN}$ 之间的锐角。τ 为控制拐点区域大小的参数，且 $\tau\in\left[1/2,1\right]^2$。$\delta_i\left(M\right)$ 表示为个体 M 在第 i 个目标上与极值点所形成的锐角。$\max\limits_{i=1.....m}\left\{\delta_i\left(M\right)\right\}+\min\limits_{i=1.....m}\left\{\delta_i\left(M\right)\right\}$ 代表着在极值点干预的情况下由个体 M 所能支配的区域的大小。$f_i\left(M\right)$ 是个体 M 在第 i 个目标上的目标函数值。ε 是一个非常小的正常数，以确保分母不为 0。N_{id} 作为理想点，由公式 2.4 定义：

$$f_j\left(N_{id}\right)=\min f_j\left(E\right)-\varepsilon \tag{2.3}$$

其中 $E=\left\{E_i\left|i=1,2,...m\right.\right\}$ 是极值点组成的集合，ε 跟上式相同均为一个极小的正常数。

图 2.3 给出了拐点支配的相关示例。其中 M 和 N 为种群中的两个个体，E_1, E_2, E_3 为每个目标维度上的极值点。ϕ 代表向量 $\overrightarrow{N_{id}M}$ 和 $\overrightarrow{MN}$ 所形成的锐角。$\delta_2 = \max\{\delta_i(M)\}$ 和 $\delta_3 = \min\{\delta_i(M)\}$ 表示当 ϕ 小于 $\tau(\delta_2 + \delta_3)$ 时，个体 M 拐点支配个体 N。此外，个体 M 和 N 在二维空间中支配区域如图 2.4 所示，阴影和虚线代表不同个体对应的支配区域。

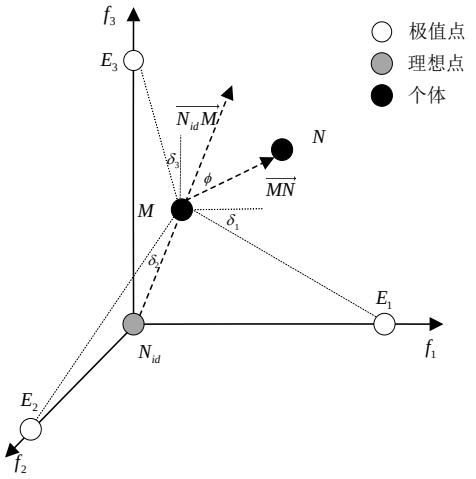


图 2.3 拐点支配关系示例

Fig. 2.3 Example of knee-oriented dominance

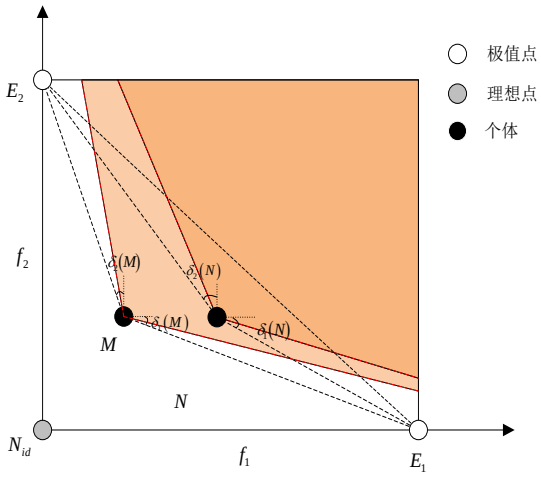


图 2.4 个体 M 和 N 所支配的区域

Fig 2.4 Dominance region of individuals M and N

2.2.2 KnMAPIO 算法的主要框架

KnMAPIO 的总体框架如算法 2.1 所示。首先，种群 P 和外部归档集 A 被初始化，同时初始化种群中每个鸽子个体的速度 V_i 与位置 X_i。然后，计算现有种群的局部中心点 P_{center} 以及每个鸽子个体 p_i 的目标函数值。接着对种群中的个体进行非支配排序并将优势个体放到外部归档集 A 中，对现有归档集进行更新。然后执行主要的进化过程。通过在 2.2.4 节中所提的速度位置更新策略来对现有的种群进行速度位置更新操作，得到

新的种群 P 并计算其适应度值，同时得到种群的局部最优个体 $pbest_i$ 和局部中心个体 P_{center}。接着对由种群 P、局部最优个体 $pbest_i$、局部中心个体 P_{center} 和外部归档集 A 所组成的父代种群 Q 进行 2.2.3 节所提的环境选择策略从而得到新的解集性能更好的种群。然后，通过在外部归档集上执行模拟二进制交叉(SBX)和基于多项式的变异(PM)两种进化策略，得到一个新的种群 S。直到循环迭代结束，得到最后的种群 P 和外部归档集 A。

Algorithm 2.1 KnMAPIO 的总体框架

开始

初始化种群 P，构建外部归档集 A；

for $i=1$ to N

初始化个体速度 V_i，位置 X_i；

计算种群中个体 p_i 的适应度值，局部最优位置 $pbest_i$；

计算种群的局部中心点 P_{center}；

end for

利用现有种群更新外部归档集 A，并计算归档集中个体适应度值；

while $T \le T_{\max}$

for $i=1$ to N

根据 2.2.4 节速度位置更新策略更新个体的速度度 V_i，位置 X_i；

计算种群中个体 p_i 的适应度值，更新局部最优位置 $pbest_i$；

end for

计算种群的局部中心点 P_{center}；

$Q=(P,P_{center},P_{best},Archive,R)$；

对父代 Q 执行 2.2.3 节的环境选择策略得到种群 P；

对外部归档集进行更新 A，并对其执行进化策略得到新种群 S；

计算新种群 S 中个体的适应度值;

更新外部归档集 A；

end while

输出：种群 P，外部归档集 A

结束

2.2.3 基于拐点支配的环境选择策略

本节主要详细介绍了基于拐点支配的环境选择策略。在 Pareto 支配的最后一层采用拐点支配策略避免了过早收敛，改善了 Pareto 支配策略导致的选择压力不足，从而提高算法的整体选择能力。图 2.5 为基于拐点支配的环境选择策略示例。整个解空间被参考向量 R_1 和 R_2 分解为两个独立的子空间 S_1 和 S_2。经过 Pareto 支配排序后个体被划分为三层 L_1，L_2 和 L_3。对划分后的最后一层个体采用拐点支配策略进行个体选择生成新的子

代。假设 A,B,C,D 为 L_3 层上的拐点。其中，根据 2.2.1 节中的拐点定义，在子空间点 S_1 中点 A 拐点支配点 B，子空间 S_2 中点 C 和点 D 之间是互相不支配的。因此，点 A、C 和点 D 作为非支配个体被选择进入新的子代。环境选择整体过程详见算法 2.2 和 2.3。

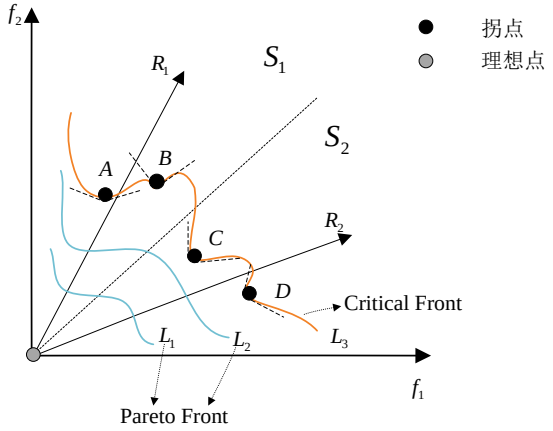


图 2.5 基于拐点支配的环境选择策略

Fig. 2.5 Environment selection strategy based on knee-oriented domination

基于拐点的环境选择过程如算法 2.2 所示。首先，子代 P 被初始化为空集。其次，对父代进行 Pareto 非支配排序划分层级 $L_1,L_2,\cdots L_l$。最后一层 L_l 之前所有的个体均被放到子代 P 中。然后对最后一层的个体进行拐点支配排序（参考算法 2.3），最后一层中的 $n-|P|$ 个个体通过拐点选择进入到子代 P 中。

算法 2.2 基于拐点的环境选择策略

开始

初始化种群 P，参考点集 R，极值点集 E；

非支配排序 $(Q,R_r)=\{L_1,L_2,\cdots\}$；

for each $L_l\in\{L_1,L_2,\cdots\}$

if $|P|+|L_l|\le n$ then

$P=P\cup L_l$

else

对最后一层个体进行拐点支配选择；

$P=P\cup KDSelection(L_l,n-|P|,E,R_r)$；

end if

end for

输出：种群 P；

结束

算法 2.3 详细介绍了算法 2.2 中使用的拐点选择策略。首先进行初始化操作。然后，将解空间划分为不同的子空间 ℓ_i 并将其设置为空集。在每个子空间中将最后一层的个体与距离最近的参考向量 R_r 相关联。对于每个子空间 C_i 中的个体，采用拐点支配关系进行排序并划分层级。然后根据拐点支配层级将不同子空间中的个体重新组合到新的集合 u 中。然后按照类似于 NAGA-II[50]中非支配排序过程，对临界层的所有解按层数升序排序，层数相同的解按拥挤距离降序排序，直到选择的解的个数得到种群大小 w，输出种群 P。

算法 2.3 拐点选择策略

开始

初始种群 W，种群大小 w，参考点集 R_r，极值点集 E；

$u=\{\ell_1,\ell_2,\cdots\}\wedge\ell_1=\varnothing,\ell_2=\varnothing,\cdots$ //* u 中的一组空列表.*//

$C:\{C_1,C_2\cdots C_k\}=Grouping(R_r)$

for each $C_i\in C$ do

执行拐点选择策略 $C_i=\{S_1^i,S_2^i,\cdots\}$

$u=\{S_1^i\cup\ell_1,S_2^i\cup\ell_2,\cdots\}$

end for

for each $\ell_i\in u\wedge|P|<l$ do

if $|P|+|\ell|\le l$ then

$P=P\cup\ell_l$

else

$P=P\cup CrowdingDistance(\ell_l,l-|P|)$

end if

end for

输出 P；

结束

2.2.4 结合不同分布策略的速度位置更新公式

本节介绍了一种新的速度位置更新策略，利用高斯分布[51]、柯西分布[52]、Levy 分布[53]的特性来提高算法的整体搜索能力。其中，柯西分布因为其尾部较粗较长，算法产生的子代更分散，子代种群的多样性更好，更适合全局优化。因此，我们在搜索过程中引入柯西分布，以提高算法在早期迭代过程中的全局搜索能力。结合 Levy 分布的搜索在随机行走过程中能够有较高的概率实现大跨度，从而保证了行走不局限于小的局部区域，可以增加种群的多样性，扩大搜索范围。这在种群的后期迭代中较高概率提供其他

的搜索方向。高斯分布可以在单个局部搜索区域内大概率小范围的变化。因此，在搜索过程中引入高斯分布，能够提高算法的局部搜索能力，这部分的速度更新因子也影响整个算法迭代过程。改进后的速度更新公式如下所示：

$$V_i\left(t_{now}\right)=e^{-Rt_{now}}\cdot V_i\left(t_{now}-1\right)+r_1\cdot Cauchy\cdot tr\cdot\left(1-\log_T^{t_{now}}\right)\cdot\left(X_{glo}-X_i\left(t_{now}\right)\right)+$$
$$r_3\cdot Levy\cdot tr\cdot\log_T^{t_{now}}\cdot\left(X_{cen}-X_i\left(t_{now}\right)\right)+r_5\cdot G\cdot\left(X_{glo}-X_{cen}\right) \tag{2.4}$$

$$X_i\left(t_{now}\right)=X_i\left(t_{now}-1\right)+V_i\left(t_{now}\right) \tag{2.5}$$

其中t_{now}表示当前迭代次数，T代表最大迭代次数，R为地图与指南针算子，tr是确保从地图与指南针算子平稳过渡到地标算子之间的迁移因子。$Cauchy$，$Levy$,和G分别表示柯西分布、Levy 分布和高斯分布因子。

X_{glo} 为个体全局最优位置，X_{cen}为当前迭代中某些个体的中心位置信息，可根据公式(2.6)计算。

$$X_{cen}=\frac{\sum_{j=1}^{n_1^x}S_{1j}^X}{n_1^x}S_1^X n_1^X \tag{2.6}$$

其中 $\sum_{j=1}^{n_1^x}S_{1i}^X$ 表示非支配集合中个体的所有解的和 S_1^X， n_1^X 是集合S_1^X中解的个数。r_1，r_3 和r_5表示三个学习因子，定义如下：

$$r_i=\begin{cases}0,0<rand()\le\dfrac{1}{M}\\ 1,\dfrac{1}{M}<rand()\le 1\end{cases} \tag{2.7}$$

其中$rand()$位于$[0,1]$之间的随机数。M代表目标空间的维度。当 r_i 为 0 时，公式 2.5 中该部分将不再影响速度的更新。当r_i是 1 时，随着目标维度M的更新，$V_i\left(t_{now}\right)$也将动态更新。

2.2.5 KnMAPIO 时间复杂度分析

本节主要分析所提的 KnMAPIO 算法的时间复杂度。该算法的计算复杂度主要来源于基于拐点支配的环境选择策略、速度位置更新策略和归档集更新策略。基于拐点支配的环境选择策略包括 Pareto 非支配排序和拐点支配排序。当种群大小为N，目标空间维度为M时，非支配排序的计算复杂度为$O\left(N^2M\right)$。拐点支配机制应用到 Pareto 非支配排序分层后的最后一层个体上，对任意两个个体，计算两个个体之间夹角的时间复杂度为$O\left(N\right)$。最坏的情况是所有的解都在同一层，这种情况下拐点支配排序的计算复杂度为$O\left(N^2M\right)$。本文采用的拥挤度机制类似于 NSGA-II，因此它的时间复杂度为$O\left(M\times N\log N\right)$。速度位置更新策略的时间复杂度为$O\left(M\right)$。归档集更新的计算复杂度

主要是通过将种群中的个体与归档集中的精英个体进行比较得到的。因为归档集的大小通常是根据种群的大小来设定的，因此最差情况下它的时间复杂度为$O\left(N^2M\right)$。综上所述，当处于最差情况下，KnMAPIO 算法的整体时间复杂度为$O\left(N^2M\right)$。

2.3 仿真实验

本节主要包括两类对比实验。第一类仿真实验在面向拐点的基准测试集 PMOPs 上对所提出的 KnMAPIO 算法和其他 4 种算法进行测试，验证了所提算法对拐点和拐点周围区域解的识别能力。第二类仿真实验在标准测试集 DTLZ 和 WFG 上与五种经典的高维多目标优化算法和三种最新提出的算法进行了比较，通过衡量这些算法获得的解在真正的 Pareto Front 中占支配地位的程度，验证了所提算法逼近真实 Pareto 前沿的能力。

2.3.1 Benchmark 测试集

本章采用两类基准测试集：衡量拐点识别能力的 PMOPs 测试集和衡量算法逼近真实 Pareto 前沿的 DTLZ 和 WFG 测试集。其中 PMOPs 测试集[37]作为评估算法精确识别拐点能力的标准测试集，具有复杂的拐点区域，测试问题具有线性、多模态、不可分解、复杂 PoF 等特性。本章选取其中的 PMOP1 - PMOP3、PMOP5 -PMOP6、PMOP8 - PMOP9 和 PMOP11 - PMOP14 作为测试用例来衡量算法的性能。DTLZ[38]和 WFG[39]是专门用来评估高维多目标优化算法能力的标准测试集，其中的测试函数具有线性、多模态、非连通、混合凸/凹和欺骗问题等明显属性。本文选择 DTLZ1-DTLZ7 和 WFG1-WFG9 问题作为衡量算法逼近真实 Pareto 前沿能力的测试问题。上述测试集的特性分析如表 2.2 所示。其中，只有 PMOPs 问题集中存在单模态问题，而且大多数的 PMOPs 测试示例具有复杂的 PoF，这使得我们所设计的仿真实验能够更好的贴合实际工程应用。

表 2.2 测试集特性分析

Table 2.2 Properties analysis of test functions

特性	测试函数
线性	PMOP1，PMOP5，PMOP7，PMOP10，PMOP13，PMOP14，DTLZ1，WFG3，DTLZ8
凹面性	PMOP2， PMOP4，PMOP8，PMOP11，DTL2，DTL3，DTLZ4，WFG4-WFG9
单模态	PMOP1，PMOP2，PMOP5，PMOP6，PMOP7，PMOP9，PMOP11，PMOP13
不可分解性	PMOP1，PMOP5，PMOP7，PMOP9，PMOP10，PMOP11， WFG2，WFG3，WFG6
复杂的 PoF	PMOP1，PMOP2，PMOP3，PMOP4，PMOP5，PMOP7，PMOP8，PMOP10，PMOP11，PMOP12

可分解性	PMOP2，PMOP3，PMOP4，PMOP6，PMOP8，PMOP12，PMOP13，PMOP14
多模态	PMOP3，PMOP4，PMOP8，PMOP10，PMOP12，PMOP14，DTLZ1, DTLZ3, DTLZ4
不可微分性	PMOP4，PMOP5
凸面性	PMOP3，PMOP6，PMOP9，PMOP12，WFG2
独特拐点区域	PMOP6，PMOP9
退化拐点区域	PMOP13，PMOP14
偏差性	DTLZ4, WFG1,WFG5，WFG7, WFG8, WFG9
连通性	WFG2, WFG3, WFG6, WFG8, WFG9
退化性	DTLZ5, DTLZ6, DTLZ7, WFG3

2.3.2 参数设置及评价指标

（1）测试集参数设置

针对 PMOPs 测试问题中的所有参数均列于表 2.3 中。其中，M 代表目标个数，N 代表决策变量个数. $k(x)$ 拐点区域函数，(A,B,s,p,l) 是一组控制拐点区域形状和数量的参数。其中具体参数的意义详见文献。DTLZ 和 WFG 测试问题中的参数设置如表 2.4 所示。

对于不同的目标空间维度，所对应的种群大小也随之改变。表 2.5 介绍了在上述两类测试问题中针对不同的目标个数，我们相应设置的种群大小。其中，对于 PMOPs 测试集我们的目标个数设置为 3，5，8，10。对于 DTLZ 和 WFG 测试集，我们的目标个数为 4，6，8，10。仿真实验中每个算法对于每个测试函数均独立运行 30 次。

表 2.3 PMOPs 测试集的参数设置

Table 2.3 Parameter setting for the PMOPs test functions

测试问题	目标个数(M)	决策变量个数(N)	$k(x)(A,B,s,p,l)$ 中参数	凸区域拐点个数
PMOP 1	3,5,8,10	$m+9$	(4, 1, -1, 1)	$\lfloor A/2 \rfloor^{m-1}$
PMOP 2,3,8,11,12	3,5,8,10	$m+9$	(4, 1, 2, 1)	$\lfloor A/2 \rfloor^{m-1}$
PMOP 5	3,5,8,10	$m+9$	(1, 1, 2, 1, 12)	$\lfloor 2*A \rfloor^{m-1}$
PMOP 6,9	3,5,8,10	$m+9$	(2,1,2,1)	$\lfloor A-1 \rfloor^{m-1}$
PMOP 13	3,5,8,10	$m+9$	(2,1,-2,1)	∞
PMOP 14	3,5,8,10	$m+9$	(2,1,-1,1)	∞

表 2.4 DTLZ 和 WFG 测试集参数设置

Table 2.4 Parameter setting for the DTLZ and WFG test functions

测试问题	决策变量个数（N）	参数	最大迭代次数
DTLZ 1	$M-1+k$	$k+5$	700
DTLZ 2	$M-1+k$	$k=5$	250
DTLZ 3	$M-1+k$	$k=5$	1000
DTLZ 4-DTLZ 6	$M-1+k$	$k=5$	250
DTLZ 7-DTLZ8	$M-1+k$	$k=20$	250
WFG 1	$k+l$	$k=M-1,l=20$	1000
WFG 2	$k+l$	$k=M-1,l=20$	700
WFG 3-WFG 9	$k+l$	$k=M-1,l=20$	250

表 2.5 目标空间维度对应的种群规模

Table 2.5 Population size corresponding to the objective space dimension

目标空间维度	种群规模
3	105
4	120
5	126
6	132
8	156
10	275
15	135

（2）测试算法参数设置

在第一类仿真对比实验中，包括 KnEA [40], LA-MOEA [41], LBD-MOEA [49], MAPIO [35]在内的四种算法跟我们所提的 KnMAPIO 在识别拐点能力方面进行了比较。所有参数均按照原文建议进行设置。其中,在 KnEA [40] 中拐点比率设置为 0.5。在 LA-MOEA[41]中，局部α支配的参数设置为$\alpha=0.75$。在 LBD-MOEA 算法中，当生成对应目标个数为 3、5、8 和 10 的参考向量时，参数(H_1,H_2)分别设置为(1, 5), (1, 3),(1, 2),和(1, 3)，其他参数详见文献[49]。对于所提的 KnMAPIO 算法,拐点支配策略中的参数τ的设置同 LBD-MOEA 算法中类似. 同时为了进行比较，我们设置了与 MAPIO[35]相同的参数, 例如迁移因子tr设为 1, 地图和指南针算子R设置为 0.3.

第二类仿真实验在标准测试集 DTLZ 和 WFG 上进行。其中，首先跟包括 NSGA-III [8], GrEA [9], MOEA/D [54], RVEA [55],和 VaEA [56]在内的五种经典算法进行了比较，同时也跟最近三年所提的包括 CSEA [57], hpaEA [58], 和 MaPSO-MC [59]在内的三种算法进行了比较。其中，CSEA 是由 Pan 等人[57]在 2019 年提出的，他们通过使用人工神经网络来预

测候选解和参考解之间的优势关系，而不是判断她们跟目标函数值的逼近程度。Tian 等人[58]在 2020 年提出了 hpaEA 算法，该算法是第一个通过将 Pareto 最优前沿表现为突出解并利用其邻近解形成的超平面来区分非支配解的算法。MaPSO-MC 是 Hu 等人[58]在 2021 年提出的，利用基于世代的适应度评价准则结合多样性增强(GBFE-DE)策略和 ISDE+策略来评估个体性能从而对混合推荐模型进行求解。

DTLZ 和 WFG 测试集中的所有算法采用的参数均参考原始论文进行设置。其中，文献 [55, 56, 57]采用的模拟二进制交叉[8]和多项式变异[8]产生子代，而交叉和变异的分布指标设置为$n_c=20$和 $n_m=20$。交叉和变异的概论设置为 $p_c=1.0$和 $p_m=1/D$，其中 D 表示决策变量的数目。对于 GrEA，对于参数每个维度中划分个数 div 的设置来源于文献[9]。MOEA/D 中的邻域个数设置为$N/10$，而其他相关参数参照[54]设置。在文献[55]中，惩罚函数的变化率 α 设为 2, 自适应参考向量变化频率为 $f_r=0.1$。对于 CSEA 算法，训练 FNN 的最大时间 T 为 500, 隐藏神经元个数 H 为 10, 参考解的个数为 6。突出解决方案的数量 κ 在 hpaEA[58]中设为 6, 其他参数的设置类似于 NSGA-III 算法。文献[59]中，学习因子 c_1,c_2,c_3 介于区间$[1.5,2.5]$中，以及影响因子θ的的数量在 MaPSO-MC 算法中设置为 2。

本文进行的所有仿真实验均是在 MATLAB 2016b 软件中由 Tian 等人[60]所提出的 PlatEMO 平台上运行的。同时硬件设施采用 64-bit Microsoft 10 的操作系统，Intel Xeon Gold 5222CPU @ 3.80 GHz 的处理器，以及 64 GB 的内存。

（3）评价指标

为了衡量本章提出的算法 KnMAPIO 和其他算法在标准测试集 PMOPs 上识别拐点以及个体选择的能力，本文使用了两个指标：拐点驱动代际距离(KGD)[37]和拐点驱动反向代际距离(KIGD)[37]。而为了进一步衡量算法在 DTLZ 和 WFG 测试集上与真实 Pareto 前沿的逼近程度，本文采用了覆盖率 Coverage[54]指标。

KGD 表示算法解集与 Pareto 前沿拐点区域中参考点的接近程度，从而评价算法的收敛性能。KGD 值越小，收敛性能越好。KGD 可以根据以下公式计算：

$$KGD=\frac{1}{|\mathbb{Q}|}\sum_{i=1}^{|\mathbb{Q}|}d\left(v_i,\mathbb{R}\right) \tag{2.9}$$

其中$\mathbb{Q}$为由算法得到的近似解集，$\mathbb{R}$表示一个在拐点区域均匀分布的参考点集，$d\left(v_i,\mathbb{R}\right)$ 表示属于近似解集$\mathbb{Q}$中的点 $\boldsymbol{v}_i$ 与集合$\mathbb{R}$中距离最近的参考点之间的欧氏距离。

KIGD 表明由算法得到的解覆盖膝关节区域的程度。KIGD 值越小，多样性能越好，也代表该算法得到的解集可以广泛覆拐点区域。 KIGD 的计算如下：

$$KIGD = \frac{1}{|\mathbb{R}|}\sum_{i=1}^{|\mathbb{R}|} d\left(v_i, \mathbb{Q}\right) \tag{2.10}$$

其中$\mathbb{Q}$和$\mathbb{R}$代表的意义同2.9式中相同，$d\left(v_i, \mathbb{Q}\right)$代表从属于参考点集$\mathbb{R}$的个体$v_i$到近似解集$\mathbb{Q}$中的点的最短欧氏距离。

Coverage指标代表解集与Pareto 前沿的支配关系，可以衡量解集的覆盖性能。Coverage计算如下：

$$C\left(Pop, PF\right) = \frac{\left|\left\{u \in PF \,\middle|\, \exists v \in Pop : v \text{dominance} u\right\}\right|}{|PF|} \tag{2.11}$$

其中*PF*代表Pareto前沿，*Pop*表示群体中所有个体的目标值。分子代表Pareto前沿中至少被种群*Pop*的一个个体所支配的解的个数。分母表示在*PF*中解的总体数。

2.3.3 在基于拐点的测试集上的实验结果分析

在本节中，表2.6和表2.7分别给出了KnEA[40]、LA-MOEA[41]、LBD-MOEA[49]、MAPIO[35]和提出的KnMAPIO在PMOPs测试集上的比较结果。表2.6给出了PMOPs测试问题中5种算法的KGD值，从而对算法的收敛性能进行评估。表2.7给出了五种算法对应的KIGD值评价结果，可以对算法的收敛性能进行评价。在这些实验结果表中，加粗黑体显示的值表示算法求得的最优解，“+”、“-”和“=”表示其他算法性能优于、劣于或者等于所提的KnMAPIO算法。从表2.6可以看出，KnEA、LA_MOEA、LBD_MOEA、MAPIO和KnMAPIO分别产生了0、9、7、8、20个最佳结果。从表2的最后一行可以看出，与KnEA相比，KnMAPIO在37个测试函数上具有明显的优势。LA_MOEA的性能与LBD_MOEA相似，其中有15项优于KnMAPIO算法。但是LA_MOEA在21个测试问题上比KnMAPIO差，LBD_MOEA在23个测试问题中比KnMAPIO差。在与现有的MAPIO算法的比较结果中可以看出，MAPIO只在7个测试问题中比KnMAPIO好，而有18个比本文算法差。这些比较结果均表明，我们的算法在KGD中具有一定的优势。从表2.6在不同PMOPs问题上算法的求解性能可以看出，KnMAPIO算法在PMOP2、3、5、6、9、11、13和14这些测试函数上具有良好的收敛性能，表明该算法在解决具有如凹面性，多模态、不可分等特性的问题上的优越性能。对于PMOP3测试问题，所提算法在四个不同的目标上都取得了良好的性能。而对于PMOP1，KnMAPIO算法在性能相对差一点，表明我们的算法在求解线性基本形状问题上需要我们进一步的改进。但在与MAPIO的比较结果中，所提的KnMAPIO算法具有明显的优势。主要原因在于该算法在原有的 BFE 策略基础上所采用的基于拐点支配的环境选择策略来提高对拐点区域个体识别的概率，以及对算法整体选择压力的提升。

表 2.6 PMOPs 测试问题中五种算法在不同目标下的 KGD 值

Table 2.6 The KGD value of five algorithms for different objectives in the PMOP test problems

Problem	M	D	KnEA	LA_MOEA	LBD_MOEA	MAPIO	KnMAPIO
PMOP1	3	12	1.8469e-3 (2.44e-4) =	1.4322e-3 (1.35e-4) +	**1.4016e-3 (1.19e-4) +**	1.9981e-3 (3.36e-4) =	1.9213e-3 (2.06e-4)
	5	14	2.2393e-2 (2.79e-3) +	**1.6326e-2 (1.38e-3) +**	1.6990e-2 (1.33e-3) +	4.4279e-2 (7.95e-3) -	3.3608e-2 (3.59e-3)
	8	17	7.7401e-2 (2.00e-2) +	**5.3710e-2 (4.16e-3) +**	5.5873e-2 (4.30e-3) +	1.8405e-1 (4.14e-2) =	1.8334e-1 (1.78e-2)
	10	19	9.6444e-2 (6.09e-2) +	**6.1521e-2 (5.47e-3) +**	6.2354e-2 (6.76e-3) +	1.6543e-1 (8.11e-2) +	2.3085e-1 (3.96e-2)
PMOP2	3	12	1.9565e-3 (2.90e-3) -	3.1685e-3 (9.44e-4) -	2.9804e-3 (8.07e-4) -	**7.0534e-4 (4.91e-5) +**	7.6479e-4 (3.33e-5)
	5	14	1.8891e-2 (9.17e-3) -	5.7350e-3 (5.81e-4) -	6.1427e-3 (7.61e-4) -	1.1061e-1 (4.73e-1) -	**5.2127e-3 (1.35e-4)**
	8	17	2.8814e+0 (1.26e+0) -	1.2864e-2 (2.02e-3) -	1.2421e-2 (2.69e-3) =	**1.1515e-2 (1.62e-3) =**	1.1769e-2 (9.64e-4)
	10	19	3.2156e+0 (2.12e+0) -	2.0847e-2 (9.18e-3) -	1.8909e-2 (5.68e-3) -	1.3402e-2 (1.77e-3) =	**1.2415e-2 (1.81e-3)**
PMOP3	3	12	9.3679e-2 (1.39e-1) -	1.4563e-2 (1.49e-2) -	1.3153e-2 (1.07e-2) -	2.8556e-3 (5.07e-3) -	**2.0614e-3 (2.36e-3)**
	5	14	3.0432e+0 (1.07e+0) -	1.5511e-2 (1.05e-2) -	1.6246e-2 (8.44e-3) -	6.3091e-2 (2.58e-1) -	**6.3105e-3 (1.41e-3)**
	8	17	7.1034e+0 (1.90e+0) -	1.3408e-2 (7.58e-3) =	1.2494e-2 (4.78e-3) =	1.3903e+0 (2.96e+0) -	**1.1617e-2 (3.31e-3)**
	10	19	5.0461e+0 (2.71e+0) -	1.0155e-1 (4.52e-2) -	1.0363e-1 (4.98e-2) -	1.3361e+0 (2.10e+0) -	**7.0070e-2 (1.30e-1)**
PMOP5	3	12	3.3007e+0 (8.58e+0) =	**7.2112e-1 (9.60e-1) =**	1.0358e+0 (8.92e-1) =	3.8080e+3 (1.41e+4) =	1.1629e+0 (1.90e+0)
	5	14	1.3326e+1 (2.28e+1) -	9.3691e-1 (5.21e-1) =	**9.2124e-1 (9.94e-1) =**	1.7550e+4 (3.47e+4) =	1.2684e+0 (2.09e+0)
	8	17	1.4201e+3 (1.43e+3) -	2.1125e+0 (1.93e+0) -	1.5023e+0 (1.23e+0) =	1.6264e+4 (5.02e+4) -	**9.1905e-1 (4.46e-2)**
	10	19	3.2018e+4 (9.13e+4) -	2.2011e+1 (1.15e+1) -	2.4439e+1 (1.10e+1) -	5.8693e+4 (6.72e+4) -	**1.7880e+0 (3.44e+0)**
PMOP6	3	12	1.3213e-1 (1.82e-1) -	2.3207e-2 (4.64e-2) =	4.3035e-3 (1.21e-2) =	**1.3016e-3 (2.73e-4) =**	1.5043e-3 (2.97e-4)
	5	14	3.5210e+0 (2.35e+0) -	1.1492e-1 (1.38e-1) -	5.9528e-2 (7.52e-2) -	5.5949e-1 (2.46e+0) -	**1.0083e-2 (8.70e-4)**
	8	17	5.5865e+2 (2.08e+2) -	1.0850e+0 (1.31e+0) -	5.9012e-1 (4.69e-1) =	3.7760e+1 (5.93e+1) =	**3.6419e-1 (8.06e-2)**
	10	19	8.2870e+3 (8.01e+3) -	**2.3691e+2 (3.48e+2) +**	2.7826e+2 (3.94e+2) +	1.8033e+3 (1.95e+3) =	1.6698e+3 (1.23e+3)

PMOP8	3	12	2.7807e-3 (3.44e-3) -	1.1518e-3 (1.57e-4) -	1.1338e-3 (2.18e-4) -	**5.6264e-4 (4.57e-5) =**	5.8972e-4 (2.91e-5)
	5	14	2.6941e-2 (7.77e-3) -	3.1981e-3 (3.56e-4) +	3.2131e-3 (3.49e-4) +	**3.0957e-3 (4.65e-4) +**	3.6720e-3 (1.36e-4)
	8	17	1.9409e-1 (4.40e-2) -	3.2794e-3 (7.56e-4) +	**3.1164e-3 (4.23e-4) +**	6.3988e-3 (1.28e-3) +	8.3614e-3 (3.57e-4)
	10	19	1.4787e-1 (2.45e-2) -	2.1042e-2 (8.72e-3) -	2.0419e-2 (7.14e-3) -	1.2530e-2 (6.13e-3) =	**9.6205e-3 (1.59e-3)**
PMOP9	3	12	1.9194e-2 (8.43e-3) -	1.3688e-2 (8.53e-3) -	1.1580e-2 (6.50e-3) -	4.5841e-2 (6.16e-2) -	**9.9630e-4 (7.80e-4)**
	5	14	2.8807e-1 (7.58e-2) -	1.5171e-2 (8.83e-3) -	1.6958e-2 (7.94e-3) -	2.4622e-2 (2.68e-2) -	**7.3974e-3 (5.26e-3)**
	8	17	9.5797e-1 (2.50e-1) -	**2.6375e-2 (9.77e-3) +**	2.8023e-2 (1.66e-2) +	8.7855e-2 (1.35e-1) -	6.1833e-2 (3.97e-2)
	10	19	1.5176e+0 (2.89e-1) -	8.3021e-2 (3.72e-2) +	**7.6761e-2 (2.57e-2) +**	2.8071e-1 (2.81e-1) +	5.3258e-1 (3.03e-1)
PMOP11	3	12	4.0225e-3 (4.92e-3) -	6.6403e-3 (6.89e-3) -	1.2664e-2 (1.33e-2) -	**7.3591e-4 (2.46e-4) =**	8.1330e-4 (5.06e-5)
	5	14	6.8024e-2 (3.37e-2) -	1.3109e-1 (1.03e-1) -	2.8282e-1 (3.11e-1) -	**1.1223e-2 (2.35e-3) +**	1.3067e-2 (4.65e-4)
	8	17	9.0247e+0 (3.39e+0) -	3.5381e-1 (1.68e-1) -	3.5049e-1 (1.50e-1) -	4.5320e-2 (6.83e-3) =	**4.4666e-2 (1.31e-3)**
	10	19	8.3257e+0 (4.67e+0) -	8.7800e-1 (4.76e-1) -	7.0604e-1 (3.59e-1) -	7.5371e-2 (1.76e-2) -	**5.0304e-2 (3.50e-3)**
PMOP12	3	12	1.7102e-2 (2.89e-2) -	1.9103e-4 (1.03e-4) =	7.7803e-4 (1.94e-3) =	**1.3622e-4 (4.20e-5) =**	1.7267e-4 (6.93e-5)
	5	14	4.8386e-1 (1.13e-1) -	3.2192e-3 (6.61e-3) =	2.5359e-3 (4.66e-3) -	1.1588e-1 (3.80e-1) =	**3.5954e-4 (1.35e-4)**
	8	17	4.7259e-1 (1.93e-1) -	**2.8734e-4 (1.37e-4) +**	3.6082e-4 (1.85e-4) +	3.6956e-1 (5.34e-1) =	1.3624e-3 (3.13e-3)
	10	19	1.7012e-1 (1.93e-1) -	**1.3372e-3 (4.90e-4) +**	1.3964e-3 (4.79e-4) +	1.2453e-1 (2.73e-1) =	4.1743e-2 (4.13e-2)
PMOP13	3	12	4.2588e-2 (2.72e-2) -	2.3506e-2 (1.23e-2) -	2.7068e-2 (1.83e-2) -	4.9823e-2 (1.06e-1) -	**1.7025e-3 (3.19e-4)**
	5	14	1.3874e+0 (4.57e-1) -	3.0477e-1 (2.06e-1) -	2.9364e-1 (2.21e-1) -	4.3872e-1 (5.27e-1) -	**9.7955e-2 (9.70e-3)**
	8	17	4.4765e+1 (3.14e+1) -	**1.9912e+0 (1.93e+0) +**	2.1635e+0 (1.45e+0) +	2.5893e+1 (1.89e+1) -	1.0305e+1 (1.03e+1)
	10	19	1.0412e+2 (1.61e+2) +	1.0088e+1 (4.96e+0) +	**8.9263e+0 (5.66e+0) +**	8.7474e+1 (6.42e+1) +	2.3184e+2 (1.38e+2)
PMOP14	3	12	2.6551e-2 (6.55e-2) -	3.4955e-2 (5.01e-2) -	1.7731e-2 (2.12e-2) -	1.0437e-3 (3.77e-4) =	**1.0194e-3 (2.84e-4)**
	5	14	2.2893e+0	1.8049e-1	1.6238e-1	1.4398e+0	**2.4340e-2**

	8	17	(1.51e+0) - 4.2772e+1	(1.46e-1) - 2.1939e-1	(1.92e-1) - **1.9790e-1**	(6.31e+0) - 6.8881e+1	**(3.11e-3)** 4.1384e-1
	10	19	(2.48e+1) - 4.6106e+1	(1.11e-1) + 9.5097e-1	**(1.64e-1) +** **8.8581e-1**	(5.90e+0) - 1.2108e+2	(7.81e-2) 2.4970e+1
			(5.45e+1) =	(6.28e-1) +	**(4.01e-1) +**	(1.28e+2) =	(2.31e+1)
+/-/=			4/37/3	15/23/6	15/21/8	7/18/19	

PMOPs 测试问题中五种算法在不同目标下的 KIGD 值如表 2.7 所示，KnMAPIO 在 20 个测试问题上获得了最优解，LBD_MOEA 仅获得了 10 个，KnEA 获得了 8 个，LA_MOEA 获得了 6 个，现有的 MAPIO 算法并未在任何测试问题中取得最优解决方案。上述对比结果表明，KnMAPIO 算法在分布性能上具有显著的优势。分析表 2.7 中不同 PMOP 测试问题上算法的性能可以看出，本文算法在 PMOP 2、3、5、8 和 11 个测试函数上具有较好的分布性能。对于 PMOP2 和 PMOP11，我们的算法在四个不同目标上都取得了良好的性能，表明该算法在解决具有凹面性和复杂 PoF 特征的问题时具有鲁棒性和多样性。尽管 KnMAPIO 算法在解决 PMOP13 问题上的表现存在一定欠缺，但从表 2.6 和 2.7 上的对比实验结果表明，所提的基于拐点支配的环境选择策略可以显著提高算法的识别拐点和个体选择的效果，从而进一步提高算法在个体选择以及加速收敛方面的能力。

表 2.7 PMOPs 测试问题中五种算法在不同目标下的 KIGD 值

Table 2.7 The KIGD value of five algorithms for different objectives in the PMOP test problems

Problem	M	D	KnEA	LA_MOEA	LBD_MOEA	MAPIO	KnMAPIO
PMOP1	3	12	**2.6853e-1**	5.9505e-1	5.4260e-1	6.0434e-1	2.7254e-1
			(1.18e-1) =	(1.26e-1) -	(2.34e-2) -	(1.83e-1) -	(6.58e-2)
	5	14	1.0157e+0	1.1116e+0	1.2159e+0	1.8858e+0	**1.0083e+0**
			(3.23e-1) =	(7.93e-2) -	(1.90e-1) -	(5.11e-1) -	**(2.01e-1)**
	8	17	2.6883e+0	**2.0252e+0**	2.0805e+0	3.8210e+0	2.5159e+0
			(5.57e-1) =	**(2.47e-1) +**	(2.47e-1) +	(1.05e+0) -	(5.98e-1)
	10	19	3.3227e+0	**2.4065e+0**	2.4333e+0	3.8954e+0	3.1598e+0
			(8.43e-1) =	**(5.82e-1) +**	(6.86e-1) +	(1.49e+0) -	(5.70e-1)
PMOP2	3	12	8.2361e-2	1.9643e-1	1.8918e-1	1.2218e-1	**5.1144e-2**
			(3.27e-2) -	(3.91e-2) -	(2.89e-2) -	(8.63e-2) -	**(2.77e-3)**
	5	14	2.1464e-1	2.5161e-1	2.6696e-1	2.7517e-1	**1.2601e-1**
			(3.66e-2) -	(1.95e-2) -	(9.84e-3) -	(6.38e-2) -	**(3.83e-3)**
	8	17	1.0408e+0	2.9526e-1	2.8723e-1	2.5802e-1	**1.7112e-1**
			(5.95e-1) -	(1.57e-2) -	(2.72e-2) -	(4.53e-2) -	**(8.86e-3)**
	10	19	4.2602e+0	2.5368e-1	2.3400e-1	2.4022e-1	**1.6852e-1**
			(1.36e+1) -	(6.42e-2) -	(3.19e-2) -	(3.94e-2) -	**(1.82e-2)**
PMOP3	3	12	**2.5274e-1**	5.8548e-1	5.8268e-1	8.7264e-1	3.6911e-1
			(1.05e-1) +	(1.85e-1) -	(1.48e-1) -	(4.08e-1) -	(9.86e-2)

	5	14	3.7703e+0 (1.77e+0) -	**4.1051e-1 (7.52e-2) =**	4.1513e-1 (5.60e-2) =	7.1453e-1 (3.25e-1) -	4.5518e-1 (1.38e-1)
	8	17	5.2670e+0 (2.46e+0) -	8.0183e-1 (1.05e-1) -	7.6525e-1 (1.22e-1) -	9.3674e-1 (2.51e-1) -	**6.3808e-1 (1.92e-1)**
	10	19	2.2139e+0 (6.92e-1) -	7.3272e-1 (1.70e-1) -	7.5130e-1 (1.58e-1) -	9.2550e-1 (3.80e-1) -	**4.6871e-1 (1.52e-1)**
PMOP5	3	12	**2.8170e+0 (2.13e+0) +**	3.3324e+0 (4.19e+0) +	4.7727e+0 (3.75e+0) +	4.2699e+0 (2.29e+0) =	9.5969e+0 (1.49e+1)
	5	14	1.4829e+1 (1.11e+1) -	**4.3223e+0 (2.12e+0) +**	4.6777e+0 (4.05e+0) +	7.0438e+0 (3.45e+0) =	1.0108e+1 (1.56e+1)
	8	17	3.3853e+2 (3.25e+2) -	1.5657e+1 (1.85e+1) -	8.5184e+0 (7.50e+0) =	3.9376e+1 (4.42e+1) -	**7.4689e+0 (4.63e-1)**
	10	19	6.8345e+2 (1.12e+3) -	8.8091e+1 (4.85e+1) -	1.0208e+2 (5.66e+1) -	9.3403e+1 (1.19e+2) -	**1.0832e+1 (1.14e+0)**
PMOP6	3	12	**1.7601e-1 (1.60e-1) +**	5.4568e-1 (1.47e-1) -	4.6391e-1 (1.20e-1) -	6.2970e-1 (1.49e-1) -	2.7046e-1 (5.56e-2)
	5	14	4.3831e+0 (3.47e+0) -	7.6308e-1 (2.75e-1) +	**7.3838e-1 (9.21e-2) +**	1.1036e+0 (4.67e-1) -	8.8021e-1 (1.21e-1)
	8	17	7.5155e+1 (3.37e+1) -	**1.7853e+1 (1.73e+0) +**	1.8490e+1 (1.59e+0) +	2.1558e+1 (1.77e+0) =	2.1015e+1 (2.21e+0)
	10	19	2.5178e+3 (6.14e+2) -	4.4665e+3 (8.82e+1) -	4.4723e+3 (9.92e+1) -	4.6534e+3 (1.26e+2) -	**1.9794e+3 (3.05e+2)**
PMOP8	3	12	7.0425e-2 (1.13e-2) -	1.4075e-1 (1.58e-2) -	1.4191e-1 (1.35e-2) -	8.2930e-2 (4.10e-2) -	**4.8745e-2 (3.21e-3)**
	5	14	1.3441e-1 (4.09e-2) -	1.0892e-1 (1.08e-2) -	1.1311e-1 (1.21e-2) -	1.2163e-1 (2.07e-2) -	**8.2575e-2 (4.88e-3)**
	8	17	8.8452e-1 (2.18e-1) -	9.0989e-2 (7.80e-3) +	**8.9956e-2 (8.29e-3) +**	1.0620e-1 (9.73e-3) =	1.0402e-1 (6.56e-3)
	10	19	7.6039e-1 (1.50e-1) -	2.4150e-1 (9.75e-2) -	2.4283e-1 (7.85e-2) -	1.4406e-1 (4.79e-2) -	**9.2146e-2 (7.88e-3)**
PMOP9	3	12	**1.3906e-1 (4.50e-2) +**	2.0702e-1 (1.61e-2) -	2.2717e-1 (3.69e-2) -	2.2026e-1 (8.45e-2) -	1.6904e-1 (3.37e-2)
	5	14	5.8132e-1 (1.71e-1) -	3.1323e-1 (3.14e-2) +	**3.0794e-1 (3.15e-2) +**	3.6916e-1 (1.07e-1) =	3.8989e-1 (7.01e-2)
	8	17	1.9717e+0 (5.31e-1) -	1.2528e+0 (1.56e-1) -	1.1900e+0 (1.28e-1) -	1.2311e+0 (4.38e-1) =	**1.0381e+0 (1.91e-1)**
	10	19	2.7666e+0 (7.77e-1) =	**2.1324e+0 (8.83e-1) +**	2.2090e+0 (8.58e-1) +	2.5009e+0 (8.30e-1) =	3.0383e+0 (8.21e-1)
PMOP11	3	12	1.3908e-1 (8.77e-2) -	4.3339e-1 (2.27e-1) -	4.3479e-1 (1.63e-1) -	2.8026e-1 (1.97e-1) -	**7.7811e-2 (5.72e-3)**
	5	14	5.2547e-1 (1.04e-1) -	1.1296e+0 (3.85e-1) -	1.4137e+0 (5.62e-1) -	5.1416e-1 (1.38e-1) -	**3.1333e-1 (8.19e-3)**
	8	17	3.9618e+0	2.5680e+0	2.2941e+0	7.8853e-1	**5.4073e-1**

	10	19	(9.52e-1) - 4.7537e+0 (2.20e+0) -	(6.86e-1) - 3.0703e+0 (1.01e+0) -	(6.27e-1) - 3.0707e+0 (1.07e+0) -	(1.37e-1) - 1.0975e+0 (2.18e-1) -	**(1.93e-2) 5.7539e-1 (4.82e-2)**
PMOP12	3 5 8 10	12 14 17 19	**2.4202e-2 (3.05e-2) +** 1.9901e-1 (6.55e-2) - 1.1159e-1 (3.93e-2) - 3.7249e-2 (1.58e-2) -	7.5334e-2 (1.25e-2) - 2.2920e-2 (6.16e-3) = 1.4485e-2 (2.07e-3) = 1.0771e-2 (2.31e-3) =	7.1418e-2 (9.85e-3) - 2.3061e-2 (4.56e-3) - **1.3576e-2 (1.71e-3) =** **1.0078e-2 (2.02e-3) =**	6.6945e-2 (2.24e-2) - 4.4371e-2 (2.65e-2) - 1.8497e-2 (2.18e-2) = 2.5190e-2 (9.69e-3) -	3.9892e-2 (6.91e-3) **1.9186e-2 (4.65e-3)** 1.4013e-2 (3.24e-3) 1.4129e-2 (9.13e-3)
PMOP13	3 5 8 10	12 14 17 19	**3.0682e-1 (1.41e-1) +** 3.0814e+0 (9.15e-1) - 2.8672e+1 (9.63e+0) = 8.9153e+1 (3.43e+1) +	5.7992e-1 (6.52e-2) - 2.1149e+0 (2.81e-1) = 2.3196e+1 (1.97e+0) = 7.4262e+1 (1.87e+1) +	6.0788e-1 (8.69e-2) - **2.0965e+0 (3.49e-1) =** **2.2053e+1 (2.06e+0) =** **7.3543e+1 (1.53e+1) +**	5.6996e-1 (1.19e-1) - 2.7325e+0 (9.71e-1) = 2.4960e+1 (6.15e+0) = 9.6421e+1 (1.48e+1) +	4.1210e-1 (1.29e-1) 2.1606e+0 (3.61e-1) 2.4559e+1 (7.42e+0) 1.1515e+2 (3.09e+1)
PMOP14	3 5 8 10	12 14 17 19	**1.7942e-1 (3.12e-1) +** 2.5583e+0 (1.75e+0) - 1.2453e+1 (6.48e+0) - 2.1500e+1 (1.49e+1) -	4.1954e-1 (9.96e-2) - 5.2732e-1 (1.64e-1) + 1.5843e+0 (3.46e-1) = 5.7148e+0 (1.23e+0) -	4.0260e-1 (6.51e-2) - **4.9098e-1 (1.28e-1) +** **1.3726e+0 (2.94e-1) +** 5.9720e+0 (1.45e+0) -	3.6961e-1 (9.20e-2) - 7.5403e-1 (5.47e-1) = 1.8290e+0 (7.60e-1) = 4.6215e+0 (3.41e+0) -	2.6477e-1 (7.99e-2) 5.5255e-1 (6.78e-2) 1.4863e+0 (1.95e-1) **2.2220e+0 (7.37e-1)**
+/-/=			8/30/6	11/26/7	12/26/6	1/31/12	

2.3.4 在 DTLZ 和 WFG 标准测试集上的实验结果分析

在本节中，表 2.8 和表 2.9 对 GrEA[9]、MOEA/D[54]、NSGA-III[8]、RVEA[55]、VaEA[56]、CSEA[57]、hpaEA[58]、MaPSO-MC[59]和提出的 KnMAPIO 在 DTLZ 测试函数上的覆盖性能进行了对比实验分析。表 2.10 显示了算法在 WFG 测试函数上覆盖性的对比结果。由表 2.8 可以看出，GrEA、MOEA/D、NSGA-III、RVEA、VaEA 和提出的 KnMAPIO 分别在 5、5、1、0、0 和 14 个测试问题中产生了最优解。在目标个数为 6、8、10 的 DTLZ4 问题中，我们的算法达到了与 MOEA/D 相同的结果。从表 2.8 的最后一行可以看出，与 RVEA 相比，KnMAPIO 在 28 个测试函数上具有显著的优势。NSGA-III 算法只在三个测试问题上比 KnMAPIO 算法好，GrEA 和 VaEA 有 4 项最优解，MOEA/D 在 6 个测试问题上比 KnMAPIO 算法好。从表 2.9 可以看出，CSEA、hpaEA、MaPSO-MC 和提出的 KnMAPIO 分别在 5、1、4 和 18 个测试问题中产生了最优解。本文算法在 DTLZ2、5

和 6 测试函数上都有很好的性能，再次证明了本文算法在求解凹函数和多峰问题方面的优越性能。然而，从表中也可以看出，KnMAPIO 在 DTLZ7 上的性能并不是最好，原因在该测试函数的多模态特性以及难以获得收敛解，使得现有的算法求解性能不足。表 2.9 的对比结果显示，本文算法的性能显著优于其他三种算法。部分问题的覆盖率为 0.00e+0，说明 PF 中的任何解都没有被最终种群中的个体所支配。而 VaEA 在 DTLZ7 上的覆盖率均为 1.0000e+0，说明 PF 中的所有解都被最终种群中的某些个体所支配。上述实验结果也进一步证明了所提算法向真实 Pareto 前沿的逼近程度。

表 2.8 DTLZ 测试集上六种算法在不同目标下的 Coverage 值

Table 2.8 The coverage value of six algorithms for different objectives in the DTLZ test problems

Problem	M	D	GrEA	MOEAD	NSGAIII	RVEA	VaEA	KnMAPIO
DTLZ1	4	8	4.5879e-1 (2.47e-1) -	7.3624e-1 (3.82e-1) -	4.2250e-1 (2.24e-1) -	5.8777e-1 (3.10e-1) -	5.9782e-1 (9.49e-2) -	**4.0408e-1 (7.60e-2)**
	6	10	**1.0480e-1 (6.87e-2) +**	4.4028e-1 (4.51e-1) =	2.9848e-1 (1.78e-1) +	6.2092e-1 (3.24e-1) -	2.9040e-1 (7.16e-2) +	4.5758e-1 (9.46e-2)
	8	12	**8.1838e-2 (8.28e-2) +**	2.7124e-1 (4.47e-1) +	2.1132e-1 (1.52e-1) +	6.0749e-1 (3.15e-1) -	1.2778e-1 (3.20e-2) +	2.8715e-1 (7.22e-2)
	10	14	2.2461e-1 (1.20e-1) -	4.6631e-1 (5.06e-1) =	1.8267e-1 (1.03e-1) -	5.4869e-1 (2.85e-1) -	1.7188e-1 (2.77e-2) -	**1.0364e-1 (3.77e-2)**
DTLZ2	4	13	**2.7778e-2 (1.96e-2) =**	1.4866e-1 (8.81e-2) -	1.2389e-1 (7.35e-2) -	1.3906e-1 (8.42e-2) -	3.0722e-1 (8.93e-2) -	4.0556e-2 (2.11e-2)
	6	15	2.3737e-2 (1.80e-2) -	7.6515e-2 (5.05e-2) -	7.9293e-2 (5.58e-2) -	4.4275e-2 (2.91e-2) -	2.0505e-1 (5.50e-2) -	**7.5758e-3 (8.44e-3)**
	8	17	2.7244e-1 (1.61e-1) -	5.4060e-2 (3.86e-2) -	1.5000e-1 (9.74e-2) -	3.9978e-2 (2.94e-2) -	2.0385e-1 (6.50e-2) -	**8.9744e-3 (8.52e-3)**
	10	19	1.2412e-1 (6.98e-2) -	2.6935e-2 (2.17e-2) -	2.1867e-1 (1.59e-1) -	1.6254e-1 (8.95e-2) -	3.4267e-1 (4.24e-2) -	**1.5152e-2 (1.04e-2)**
DTLZ3	4	13	1.9675e-1 (1.05e-1) =	6.9837e-1 (3.45e-1) -	**1.0463e-1 (6.17e-2) +**	4.7460e-1 (2.56e-1) -	1.3375e-1 (6.81e-2) +	2.3634e-1 (1.27e-1)
	6	15	**2.1212e-2 (2.42e-2) +**	6.0294e-1 (3.76e-1) -	1.2980e-1 (7.83e-2) =	4.7319e-1 (2.54e-1) -	1.2828e-1 (4.84e-2) +	1.7407e-1 (8.36e-2)
	8	17	**4.1239e-2 (3.22e-2) +**	4.4444e-1 (4.53e-1) =	9.1026e-2 (7.93e-2) =	4.6503e-1 (2.48e-1) -	6.3034e-2 (3.51e-2) =	7.9915e-2 (4.77e-2)
	10	19	3.4873e-1 (1.85e-1) -	3.5065e-1 (4.16e-1) -	9.9394e-2 (7.29e-2) -	4.6986e-1 (2.04e-1) -	1.2703e-1 (2.98e-2) -	**1.0667e-2 (1.02e-2)**
DTLZ4	4	13	2.0556e-2 (1.97e-2) =	**3.9313e-3 (1.56e-2) +**	7.7500e-2 (5.89e-2) -	1.1776e-1 (8.04e-2) -	2.7000e-1 (9.80e-2) -	2.8333e-2 (1.88e-2)
	6	15	5.5556e-3 (6.87e-3) -	0.0000e+0 (0.00e+0) =	3.7879e-3 (4.77e-3) -	8.9628e-3 (9.70e-3) -	1.0480e-1 (4.90e-2) -	**0.0000e+0 (0.00e+0)**
	8	17	9.8291e-3	0.0000e+0	1.4957e-3	5.8110e-3	1.6453e-2	**0.0000e+0**

	10	19	(9.33e-3) - 1.2121e-4 (6.64e-4) =	(0.00e+0) = 0.0000e+0 (0.00e+0) =	(2.76e-3) - 0.0000e+0 (0.00e+0) =	(6.64e-3) - 2.4287e-4 (9.24e-4) =	(1.38e-2) - 1.9394e-3 (2.82e-3) -	**(0.00e+0) 0.0000e+0 (0.00e+0)**
DTLZ5	4	13	2.2583e-1 (3.47e-2) -	3.4251e-1 (1.13e-1) -	5.5667e-1 (1.20e-1) -	5.8905e-1 (8.79e-2) -	2.6556e-1 (2.88e-2) -	**1.9111e-1 (3.38e-2)**
	6	15	1.3359e-1 (4.08e-2) -	3.8518e-1 (9.84e-2) -	3.6869e-1 (8.50e-2) -	3.1238e-1 (1.00e-1) -	2.7778e-1 (3.54e-2) -	**1.9444e-2 (1.03e-2)**
	8	17	1.6346e-1 (5.65e-2) -	4.4867e-1 (9.84e-2) -	3.2650e-1 (9.61e-2) -	4.9464e-1 (1.52e-1) -	4.1303e-1 (5.79e-2) -	**1.3889e-2 (8.76e-3)**
	10	19	2.0279e-1 (7.03e-2) -	5.1906e-1 (1.03e-1) -	3.7382e-1 (7.12e-2) -	5.6948e-1 (1.39e-1) -	5.1479e-1 (8.46e-2) -	**4.7273e-3 (2.17e-3)**
DTLZ6	4	13	1.6361e-1 (8.29e-2) -	2.2552e-1 (2.69e-1) -	3.8056e-1 (2.08e-1) -	4.6218e-1 (1.91e-1) -	3.8667e-1 (1.67e-1) -	**3.5278e-2 (2.49e-2)**
	6	15	3.1061e-1 (1.40e-1) -	4.6105e-1 (2.73e-1) -	7.2020e-1 (2.42e-1) -	4.0089e-1 (1.45e-1) -	8.9672e-1 (2.11e-2) -	**1.3889e-2 (8.92e-3)**
	8	17	7.7885e-1 (2.31e-1) -	5.1588e-1 (2.58e-1) -	6.2329e-1 (1.55e-1) -	6.0568e-1 (1.26e-1) -	8.7073e-1 (1.50e-2) -	**6.6239e-3 (1.17e-3)**
	10	19	7.8085e-1 (1.28e-1) -	5.4339e-1 (2.16e-1) -	5.6812e-1 (1.27e-1) -	6.8867e-1 (1.44e-1) -	9.1576e-1 (1.62e-2) -	**3.6364e-3 (1.32e-18)**
DTLZ7	4	23	7.3917e-1 (3.14e-1) -	**1.0312e-1 (6.12e-2) +**	9.5750e-1 (9.92e-2) -	9.3947e-1 (1.41e-1) -	1.0000e+0 (0.00e+0) -	1.4083e-1 (5.86e-2)
	6	25	6.5227e-1 (2.72e-1) -	**4.6063e-3 (4.78e-3) +**	9.2626e-1 (1.71e-1) -	8.5340e-1 (3.10e-1) -	1.0000e+0 (0.00e+0) -	1.2500e-1 (7.00e-2)
	8	27	6.3718e-1 (3.02e-1) -	**4.2735e-4 (1.63e-3) +**	8.7842e-1 (2.78e-1) -	8.3217e-1 (2.87e-1) -	1.0000e+0 (0.00e+0) -	1.2158e-1 (4.92e-2)
	10	29	8.3709e-1 (3.71e-1) -	**0.0000e+0 (0.00e+0) +**	8.7455e-1 (2.86e-1) -	8.3392e-1 (2.92e-1) -	1.0000e+0 (0.00e+0) -	3.7188e-1 (1.74e-1)
+/-/=			4/20/4	6/16/6	3/22/3	0/27/1	4/23/1	

表 2.9 DTLZ 测试集上四种算法在不同目标下的 Coverage 值

Table 2.9 The coverage value of four algorithms for the last years on the DTLZ test problems

Problem	M	D	CSEA	hpaEA	MaPSO-MC	KnMAPIO
DTLZ1	4	8	2.0738e-1 (8.19e-2) =	5.6677e-1 (9.87e-2) =	**1.9483e-1 (4.16e-2) +**	3.5385e-1 (1.55e-1)
	6	10	**1.2194e-1 (9.48e-2) +**	2.6597e-1 (5.13e-2) +	2.2651e-1 (8.42e-2) +	4.3182e-1 (1.13e-1)
	8	12	**1.3484e-1 (9.03e-2) +**	3.4003e-1 (4.40e-2) =	6.0640e-1 (2.34e-1) =	2.7949e-1 (7.91e-2)
	10	14	1.2299e-1 (8.45e-2) =	3.2379e-1 (5.84e-2) -	7.3257e-1 (6.57e-2) -	**1.0691e-1 (1.88e-2)**
DTLZ2	4	13	4.3457e-2 (4.55e-3) =	2.3866e-1 (4.99e-2) -	**2.1864e-2 (1.14e-2) =**	3.3333e-2 (8.33e-3)
	6	15	1.4294e-2	1.4643e-1	**0.0000e+0**	9.0909e-3

	8	17	(2.44e-3) = 6.9210e-3 (2.48e-3) =	(2.05e-2) - 3.3982e-1 (4.71e-2) -	**(0.00e+0) =** 2.5641e-2 (1.76e-2) =	(8.30e-3) **6.4103e-3 (6.41e-3)**
	10	19	**6.2743e-3 (3.62e-3) =**	4.9282e-1 (1.03e-1) -	6.6406e-2 (5.61e-3) -	1.7455e-2 (1.13e-2)
DTLZ3	4	13	1.6000e-1 (1.02e-1) =	**1.1421e-1 (1.21e-1) =**	1.1682e-1 (6.00e-2) +	2.3337e-1 (5.68e-2)
	6	15	7.7102e-2 (2.25e-2) =	6.5152e-2 (6.78e-3) =	**2.0237e-2 (1.50e-2) =**	1.3333e-1 (7.64e-2)
	8	17	**3.0030e-2 (1.76e-2) =**	1.6785e-1 (2.52e-2) =	3.7703e-1 (1.09e-1) -	8.0769e-2 (6.50e-2)
	10	19	3.6201e-2 (2.04e-2) =	2.7883e-1 (7.72e-2) -	4.6611e-1 (9.24e-2) -	**2.0364e-2 (1.38e-2)**
DTLZ4	4	13	4.0185e-2 (6.46e-3) =	7.6634e-2 (7.43e-2) =	1.8333e-2 (2.24e-2) =	**1.5000e-2 (2.07e-2)**
	6	15	4.5612e-3 (3.49e-3) -	1.8417e-2 (1.95e-2) -	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
	8	17	8.9403e-4 (8.68e-4) -	7.8836e-3 (7.69e-3) =	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
	10	19	1.2099e-4 (1.49e-4) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
DTLZ5	4	13	**1.5942e-1 (3.69e-2) =**	3.0333e-1 (4.51e-2) -	4.1731e-1 (3.99e-2) -	1.8667e-1 (3.75e-2)
	6	15	1.4225e-1 (3.11e-2) -	3.5606e-1 (4.25e-2) -	4.2605e-1 (6.48e-2) -	**2.1212e-2 (6.34e-3)**
	8	17	1.3577e-1 (3.13e-2) -	3.1923e-1 (2.85e-2) -	2.6525e-1 (1.11e-1) -	**8.9744e-3 (5.73e-3)**
	10	19	1.3757e-1 (2.50e-2) -	3.4473e-1 (3.81e-2) -	3.4531e-1 (1.16e-1) -	**4.3636e-3 (1.63e-3)**
DTLZ6	4	13	2.1035e-1 (2.81e-2) -	4.3833e-1 (1.85e-1) -	3.2251e-1 (8.11e-2) -	**4.3333e-2 (4.10e-2)**
	6	15	2.8056e-1 (3.47e-2) -	7.5748e-1 (2.16e-2) -	5.5960e-1 (1.25e-1) -	**1.0606e-2 (4.15e-3)**
	8	17	1.5116e-1 (1.84e-2) -	7.6537e-1 (4.40e-2) -	5.6244e-1 (1.60e-1) -	**6.4103e-3 (0.00e+0)**
	10	19	1.3233e-1 (2.91e-2) -	8.3200e-1 (2.92e-2) -	7.0163e-1 (7.34e-2) -	**3.6364e-3 (0.00e+0)**
DTLZ7	4	23	4.1363e-1 (6.31e-2) -	9.8833e-1 (1.83e-2) -	9.6988e-1 (4.70e-2) -	**1.2000e-1 (3.89e-2)**
	6	25	2.6336e-1 (6.63e-2) =	9.9394e-1 (9.88e-3) -	9.7871e-1 (4.26e-2) -	**2.3485e-1 (1.16e-1)**
	8	27	2.4612e-1 (5.60e-2) -	1.0000e+0 (0.00e+0) -	9.7183e-1 (5.55e-2) -	**1.0256e-1 (2.94e-2)**

	10	29	4.4943e-1 (1.51e-1) =	1.0000e+0 (0.00e+0) -	1.0000e+0 (0.00e+0) -	**4.4873e-1** **(8.48e-2)**
+/-/=			2/11/15	1/19/8	3/16/9	

表 2.10 显示了 KnMAPIO 与 5 种经典算法在 WFG 测试函数上 Coverage 值的比较结果。其中 GrEA、MOEA/D、NSGA-III、RVEA、VaEA 和所提的 KnMAPIO 分别在 9、4、0、0、0 和 19 个测试问题上产生最优解。KnMAPIO 在 WFG1 的所有目标上都取得了与其他算法相同的结果，且覆盖值均为 1.0000e+0，说明 PF 中的所有解都被种群中的某些解所支配，也证明上述算法在求解偏置问题时具有相同的性能。MOEA/D 算法在 5 个问题上优于 KnMAPIO 算法，但有近三分之二的结果劣于该算法。KnMAPIO 算法在 WFG 4、5、6 和 9 测试函数上都有非常好的性能，尤其是在 WFG5 上，也进一步证明该算法对求解凹形状函数具有最佳的求解效果。上述的结果均表明，本文算法的性能要优于其他算法，也进一步证明基于拐点支配的环境选择策略和基于不同分别的个体更新策略对提高个体的选择压力以及使其趋向于真正的 PF 的能力。

表 2.10 WFG 测试集上六种算法在不同目标下的 Coverage 值

Table 2.10 The coverage value of six algorithms for different objectives in the WFG test problems

Problem	M	D	GrEA	MOEAD	RVEA	NSGAIII	VaEA	KnMAPIO
WFG1	4	13	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)**
	6	15	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)**
	8	17	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)**
	10	19	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)**
WFG2	4	13	**6.6375e-1** **(1.48e-1) +**	9.5170e-1 (8.27e-2) =	8.6233e-1 (8.80e-2) +	9.0833e-1 (6.43e-2) +	9.2458e-1 (4.67e-2) +	9.6625e-1 (1.80e-2)
	6	15	2.0833e-1 (1.08e-1) -	3.9986e-1 (3.53e-1) -	3.6283e-1 (1.37e-1) -	3.4508e-1 (1.34e-1) -	7.0492e-1 (1.12e-1) -	**1.6288e-1** **(1.45e-1)**
	8	17	**1.4103e-2** **(1.56e-2) =**	7.0152e-2 (1.59e-1) =	8.3573e-2 (1.06e-1) -	7.2756e-2 (5.60e-2) -	4.2051e-1 (1.05e-1) -	1.6346e-2 (2.52e-2)
	10	19	2.2982e-1 (8.50e-2) +	**7.9488e-2** **(2.22e-1) +**	2.8001e-1 (1.21e-1) +	2.9600e-1 (9.08e-2) +	5.4618e-1 (1.34e-1) -	3.6727e-1 (1.13e-1)
WFG3	4	13	2.0833e-1 (4.42e-2) -	9.7711e-2 (3.76e-2) =	2.9012e-1 (4.78e-2) -	2.1667e-1 (3.05e-2) -	1.7708e-1 (3.05e-2) -	**8.0833e-2** **(2.21e-2)**
	6	15	6.4773e-2 (1.52e-2) -	**5.6818e-3** **(5.43e-3) +**	5.9463e-2 (2.44e-2) -	5.7576e-2 (6.22e-2) -	8.5227e-2 (3.18e-2) -	2.0076e-2 (7.88e-3)
	8	17	6.2500e-2 (1.50e-2) -	**0.0000e+0** **(0.00e+0) +**	2.7750e-2 (1.95e-2) -	7.7244e-2 (3.38e-2) -	5.9615e-2 (2.37e-2) -	1.2500e-2 (5.29e-3)

	10	19	1.9636e-2 (4.78e-3) -	**0.0000e+0 (0.00e+0) +**	1.8235e-2 (9.73e-3) -	6.5636e-2 (2.36e-2) -	3.7636e-2 (1.15e-2) -	8.0000e-3 (5.73e-3)
WFG4	4	13	6.4500e-1 (5.67e-2) =	9.8609e-1 (2.66e-2) -	9.4731e-1 (2.79e-2) -	9.3625e-1 (3.26e-2) -	9.5708e-1 (2.25e-2) -	**6.3625e-1 (6.78e-2)**
	6	15	3.0720e-1 (3.87e-2) -	4.8399e-1 (8.44e-2) -	4.6930e-1 (6.09e-2) -	4.7652e-1 (1.24e-1) -	4.4659e-1 (6.51e-2) -	**2.0341e-1 (4.36e-2)**
	8	17	1.7051e-1 (3.23e-2) -	6.2188e-1 (1.00e-1) -	3.3488e-1 (6.29e-2) -	2.7692e-1 (6.80e-2) -	1.9712e-1 (2.93e-2) -	**1.2853e-1 (2.18e-2)**
	10	19	**4.5636e-2 (2.06e-2) +**	7.2709e-1 (1.34e-1) -	3.1210e-1 (7.07e-2) -	9.1273e-2 (3.61e-2) =	1.1309e-1 (2.20e-2) =	1.0345e-1 (2.01e-2)
WFG5	4	13	8.4042e-1 (3.20e-2) -	9.8041e-1 (1.90e-2) -	9.7353e-1 (1.53e-2) -	9.7833e-1 (1.56e-2) -	9.7750e-1 (1.51e-2) -	**7.3167e-1 (3.18e-2)**
	6	15	2.9924e-1 (5.21e-2) -	8.0567e-1 (5.30e-2) -	5.0910e-1 (6.64e-2) -	4.7727e-1 (6.26e-2) -	4.8485e-1 (4.66e-2) -	**1.3712e-1 (2.30e-2)**
	8	17	1.7821e-1 (2.95e-2) -	9.0509e-1 (2.39e-2) -	4.5119e-1 (8.53e-2) -	3.3718e-1 (5.29e-2) -	2.4519e-1 (2.40e-2) -	**7.3397e-2 (1.11e-2)**
	10	19	1.3327e-1 (1.75e-2) -	9.1205e-1 (3.68e-2) -	5.1249e-1 (9.85e-2) -	2.9145e-1 (6.95e-2) -	1.9418e-1 (2.17e-2) -	**3.9636e-2 (1.07e-2)**
WFG6	4	13	**9.7458e-1 (3.90e-2) +**	9.9680e-1 (7.04e-3) +	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	9.9917e-1 (3.73e-3) =	1.0000e+0 (0.00e+0)
	6	15	5.9280e-1 (7.74e-2) =	9.6534e-1 (4.40e-2) -	6.8979e-1 (1.21e-1) -	7.2538e-1 (7.50e-2) -	7.0909e-1 (7.81e-2) -	**5.5530e-1 (3.62e-2)**
	8	17	3.5449e-1 (5.43e-2) -	9.6251e-1 (3.44e-2) -	6.5698e-1 (1.19e-1) -	5.5994e-1 (5.22e-2) -	3.7115e-1 (5.57e-2) -	**2.4231e-1 (3.04e-2)**
	10	19	2.2491e-1 (3.50e-2) -	9.7176e-1 (2.97e-2) -	6.0188e-1 (1.02e-1) -	4.3164e-1 (8.20e-2) -	3.2073e-1 (3.76e-2) -	**1.1436e-1 (1.52e-2)**
WFG7	4	13	**3.4333e-1 (3.40e-2) +**	9.5430e-1 (5.36e-2) -	9.3964e-1 (3.31e-2) -	9.5125e-1 (3.26e-2) -	9.6292e-1 (2.03e-2) -	3.9292e-1 (7.26e-2)
	6	15	2.2462e-1 (4.44e-2) -	5.2022e-1 (1.24e-1) -	4.5206e-1 (7.56e-2) -	5.8030e-1 (6.29e-2) -	4.9697e-1 (8.59e-2) -	**1.6856e-1 (2.84e-2)**
	8	17	1.9359e-1 (3.13e-2) -	5.9153e-1 (1.35e-1) -	3.7343e-1 (9.09e-2) -	3.4135e-1 (1.02e-1) -	2.7821e-1 (5.89e-2) -	**1.2436e-1 (2.37e-2)**
	10	19	**4.4000e-2 (7.82e-3) +**	7.0603e-1 (1.21e-1) -	3.9081e-1 (7.28e-2) -	1.7545e-1 (6.85e-2) =	1.8218e-1 (2.98e-2) -	1.5636e-1 (3.83e-2)
WFG8	4	13	**9.6208e-1 (1.68e-2) =**	9.9957e-1 (1.93e-3) -	9.9640e-1 (6.79e-3) -	9.9500e-1 (7.84e-3) -	9.9625e-1 (5.04e-3) -	9.7042e-1 (1.52e-2)
	6	15	8.3068e-1 (3.65e-2) -	9.9955e-1 (2.01e-3) -	9.3790e-1 (3.45e-2) -	8.6250e-1 (7.71e-2) -	8.6894e-1 (3.36e-2) -	**6.2652e-1 (5.08e-2)**
	8	17	7.4712e-1 (3.23e-2) -	1.0000e+0 (0.00e+0) -	8.8322e-1 (1.10e-1) -	5.5481e-1 (2.11e-1) -	7.0737e-1 (4.17e-2) -	**3.6218e-1 (4.56e-2)**
	10	19	**6.5455e-2 (4.27e-2) +**	1.0000e+0 (0.00e+0) -	7.6137e-1 (6.90e-2) -	4.8055e-1 (1.02e-1) -	6.3436e-1 (3.65e-2) -	3.9364e-1 (4.09e-2)
WFG9	4	13	**6.4167e-1**	9.9246e-1	9.8242e-1	9.8917e-1	9.7250e-1	7.4917e-1

	6	15	**(6.78e-2) =** 3.3068e-1	(1.50e-2) - 9.7253e-1	(1.87e-2) - 7.0578e-1	(1.51e-2) - 6.7727e-1	(2.89e-2) - 7.3712e-1	(1.86e-1) **2.3295e-1**
	8	17	(1.03e-1) - 2.0801e-1	(4.65e-2) - 9.7050e-1	(1.22e-1) - 5.0030e-1	(1.27e-1) - 4.3590e-1	(1.11e-1) - 3.7340e-1	**(1.53e-1)** **8.7500e-2**
	10	19	(6.03e-2) - 5.3091e-2	(4.67e-2) - 9.8128e-1	(1.40e-1) - 3.6511e-1	(1.63e-1) - 2.3382e-1	(1.05e-1) - 2.1491e-1	**(3.12e-2)** **3.1818e-2**
			(2.20e-2) -	(3.52e-2) -	(1.03e-1) -	(1.00e-1) -	(3.55e-2) -	**(1.33e-2)**
+/-/=			7/20/9	5/24/7	2/29/5	2/27/7	1/29/6	

此外，为验证基于不同分布策略对算法性能的影响，在 DTLZ 测试集上进行了相应的消融实验。其中，分别考虑仅使用一个分布，结合两种分布策略和同时结合三种分布对算法性能的影响，实验结果如表 2.11 所示。MAPIOC 中仅采用柯西分布。MAPIOG 算法只使用高斯分布。MAPIOLC 算法是同时使用 Levy 和 Cauchy 分布的算法，MAPIOGC 算法是同时使用高斯分布和 Cauchy 分布的算法。从表 2.11 最后一行可以看出，MAPIO 在 9 个测试问题上比 KnMAPIO 性能差，但没有比 KnMAPIO 性能好的解。因此，虽然 KnMAPIO 算法在某些测试问题上存在不足，但总体性能优于其他比较算法。且仅使用一种分布的算法性能略差于使用两种分布的算法性能，使用两种分布的算法性能略差于三种分布组合的算法性能。因此，所提出的结合不同分布特性的速度位置更新策略是必要且有效的。

表 2.11 DTLZ 测试问题中不同目标下消融算法的覆盖值

Table 2.11 The Coverage value of algorithms for different objectives in the DTLZ test problems

Problem	M	MAPIO	MAPIOC	MAPIOG	MAPIOLC	MAPIOGC	KnMAPIO
DTLZ1	4	5.4567e-1 (1.50e-1) =	4.5275e-1 (2.63e-1) =	5.8506e-1 (1.92e-1) =	**3.9092e-1 (1.61e-1) =**	5.7206e-1 (1.74e-1) =	4.5156e-1 (1.06e-1)
	6	**5.6970e-1 (6.63e-2) =**	5.6970e-1 (5.75e-2) =	6.6818e-1 (1.10e-1) =	7.0455e-1 (1.54e-1) =	5.9394e-1 (1.45e-1) =	6.3030e-1 (1.78e-1)
	8	6.0333e-1 (9.14e-2) =	6.3333e-1 (1.20e-1) =	5.9103e-1 (2.00e-1) =	**4.8718e-1 (1.88e-1) =**	5.1410e-1 (1.40e-1) =	5.2607e-1 (6.96e-2)
	10	6.2109e-1 (4.86e-2) -	4.5309e-1 (1.17e-1) =	4.5382e-1 (5.85e-2) =	4.6327e-1 (6.92e-2) =	4.6182e-1 (1.04e-1) =	**3.7382e-1 (8.74e-2)**
DTLZ2	4	9.5000e-2 (3.21e-2) -	**3.1667e-2 (1.09e-2) =**	3.6667e-2 (2.25e-2) =	4.5000e-2 (2.54e-2) =	3.3333e-2 (8.33e-3) =	3.1667e-2 (2.07e-2)
	6	4.6970e-2 (2.10e-2) -	1.9697e-2 (8.64e-3) =	1.0606e-2 (4.15e-3) =	7.5758e-3 (7.58e-3) =	**3.0303e-3 (4.15e-3) =**	1.0606e-2 (8.64e-3)
	8	3.3333e-2 (9.51e-3) =	2.3077e-2 (1.85e-2) =	2.5641e-2 (1.63e-2) =	1.9231e-2 (9.07e-3) =	2.0513e-2 (1.05e-2) =	**1.9231e-2 (1.20e-2)**
	10	6.9818e-2 (2.11e-2) =	8.1455e-2 (1.58e-2) -	7.6364e-2 (2.37e-2) -	6.2545e-2 (2.46e-2) =	5.0182e-2 (2.20e-2) =	**4.6545e-2 (1.90e-2)**
DTLZ	4	2.0199e-1	8.3043e-2	1.1681e-1	1.2603e-1	**7.2601e-2**	1.2543e-1

3	6	(6.44e-2) = 4.0222e-1	(6.64e-2) = 3.8485e-1	(8.69e-2) = 3.4848e-1	(7.89e-2) = 4.2121e-1	**(4.89e-2) = 2.7917e-1**	(1.18e-1) 3.1667e-1
	8	(9.77e-2) = **3.6597e-1**	(1.14e-1) = 4.0000e-1	(1.31e-1) = 3.8462e-1	(1.25e-1) = 3.9159e-1	**(1.12e-1) =** 3.9872e-1	(8.66e-2) 4.0000e-1
	10	**(9.46e-2) =** 4.2691e-1	(1.18e-1) = 4.2182e-1	(8.64e-2) = 3.8182e-1	(1.86e-1) = 3.5636e-1	(1.31e-1) = 4.0727e-1	(1.10e-1) **1.9636e-1**
		(8.06e-2) -	(1.00e-1) -	(8.88e-2) -	(7.56e-2) -	(7.47e-2) -	**(9.58e-2)**
DTLZ 4	4	7.1667e-2 (2.80e-2) -	4.0000e-2 (3.25e-2) =	**1.1667e-2 (2.61e-2) =**	3.3333e-2 (1.02e-2) =	2.5000e-2 (3.73e-2) =	2.8333e-2 (1.73e-2)
	6	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
	8	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
	10	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
DTLZ 5	4	**2.6667e-1 (8.25e-2) =**	3.1833e-1 (1.16e-1) =	3.6167e-1 (3.36e-2) =	3.7833e-1 (4.66e-2) =	3.0167e-1 (6.91e-2) =	3.4000e-1 (1.13e-1)
	6	**6.9697e-2 (3.14e-2) =**	1.3333e-1 (3.92e-2) -	1.7424e-1 (6.15e-2) -	1.1364e-1 (7.74e-2) =	1.2273e-1 (8.81e-2) =	7.4242e-2 (4.09e-2)
	8	1.9231e-2 (4.53e-3) =	2.1795e-2 (1.16e-2) =	**1.4103e-2 (8.36e-3) =**	4.2308e-2 (4.36e-2) =	3.0769e-2 (1.05e-2) =	2.6923e-2 (2.19e-2)
	10	1.0182e-2 (3.98e-3) =	**4.3636e-3 (1.63e-3) +**	1.6727e-2 (7.09e-3) =	1.2364e-2 (4.15e-3) =	1.3818e-2 (5.39e-3) =	1.5273e-2 (1.22e-2)
DTLZ 6	4	2.7333e-1 (1.49e-2) -	2.1167e-1 (1.54e-1) =	2.3667e-1 (1.37e-1) -	3.8167e-1 (3.54e-1) =	2.5333e-1 (2.30e-1) =	**6.1667e-2 (4.74e-2)**
	6	3.7424e-1 (1.85e-1) -	**3.6364e-2 (3.32e-2) =**	1.1970e-1 (1.34e-1) =	1.6970e-1 (1.50e-1) =	4.6970e-2 (3.45e-2) =	9.8485e-2 (7.93e-2)
	8	5.4744e-1 (1.66e-1) -	2.0000e-1 (1.13e-1) -	1.8718e-1 (1.35e-1) -	4.0128e-1 (1.07e-1) -	3.6282e-1 (2.66e-1) -	**3.2051e-2 (4.44e-2)**
	10	7.0255e-1 (9.28e-2) -	8.1091e-1 (6.69e-2) -	7.8109e-1 (2.96e-2) -	7.2655e-1 (8.72e-2) -	8.1891e-1 (7.34e-2) -	**2.3564e-1 (7.25e-2)**
DTLZ 7	4	**9.8500e-1 (3.35e-2) =**	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	1.0000e+0 (0.00e+0) =	9.9667e-1 (7.45e-3)
	6	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)**
	8	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)**
	10	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)**
+/-/=		0/9/19	1/5/22	0/6/22	0/3/25	0/3/25	

为验证算法所提不同策略对性能的影响，在 DTLZ1-3 测试示例上进行了相应的消融实验，如表 2.12 所示。其中 KnMAPIO_noEnviroment 算法仅使用基于不同分布的种

群更新策略，KnMAPIO_nodistribution 算法仅使用基于拐点支配环境选择策略。由对比实验结果可以看出，所提的基于拐点支配的环境选择策略对算法性能的提升相对重要，单独使用一种策略的算法性能要差于所提的 KnMAPIO 算法，同时实验结果也验证了所提的两种策略对算法性能提升的重要作用。

表 2.12 DTLZ 测试问题中不同目标下不同策略消融算法的 IGD 值

Table 2.12 The IGD value of different strategies for different objectives in the DTLZ test problems

Problem	M	D	KnMAPIO_noEnvironment	KnMAPIO_nodistribution	KnMAPIO
DTLZ1	4	8	**2.2865e+0 (7.76e-1) =**	2.9545e+0 (1.31e+0) =	3.1591e+0 (1.20e+0)
	6	10	2.5599e+0 (1.08e+0) =	**2.4759e+0 (1.15e+0) =**	2.4969e+0 (8.04e-1)
	8	12	3.7462e+0 (9.79e-1) =	5.0489e+0 (2.02e+0) =	**3.4867e+0 (8.13e-1)**
	10	14	**6.9550e+0 (2.16e+0) =**	7.9030e+0 (1.80e+0) =	7.2111e+0 (2.03e+0)
DTLZ2	4	13	1.5041e-1 (3.92e-3) -	1.4604e-1 (3.35e-3) =	**1.4562e-1 (2.93e-3)**
	6	15	2.7629e-1 (2.58e-3) -	2.7045e-1 (1.64e-3) =	**2.6919e-1 (2.92e-3)**
	8	17	3.7173e-1 (1.65e-2) =	**3.5996e-1 (2.43e-3) =**	3.6188e-1 (2.44e-3)
	10	19	4.4176e-1 (7.92e-3) -	4.1808e-1 (4.88e-3) =	**4.1485e-1 (2.47e-3)**
DTLZ3	4	13	1.2492e+2 (1.25e+1) -	1.2534e+2 (1.07e+1) -	**1.1658e+2 (1.39e+1)**
	6	15	**1.1404e+2 (8.52e+0) =**	1.1921e+2 (1.27e+1) -	1.1549e+2 (1.97e+1)
	8	17	1.3592e+2 (2.26e+1) -	**1.2952e+2 (1.66e+1) =**	1.3688e+2 (2.07e+1)
	10	19	1.7229e+2 (6.91e+0) -	1.6626e+2 (8.07e+0) -	**1.4713e+2 (2.19e+1)**
+/-/=			0/6/6	0/3/9	

2.4 本章小结

本章首先对现有的高维多目标鸽群算法存在的个体选择压力不足的问题进行了分析，并介绍了现有的支配策略。然后针对随着目标急剧增加导致的 Pareto 支配选择压力不足以及获得的解集无法逼近真实 Pareto 前沿的问题，设计了一种基于拐点支配的高维多目标鸽群算法 KnMAPIO。其中，基于拐点支配的环境选择策略提高算法对拐点及其周围区域解的识别能力，进一步提高算法的个体选择压力。结合高斯分布、柯西分布和 Levy 分布的种群更新策略，能够在迭代的不同阶段动态调整搜索策略，从而提高算法的整体搜索性能。最后，通过对 PMOPs 测试集、DTLZ 和 WFG 测试集上的两种仿真对比实验的分析也证明了所提的 KnMAPIO 算法与其他算法相比对个体的选择能力更强，也更加的逼近真实 Pareto 前沿。

第三章 基于 KIGD 指标的竞争高维多目标鸽群算法

第二章从个体选择压力的角度对高维多目标鸽群算法进行分析，较好的解决了在目标个数增加时选择压力不足的问题。本章则首先通过分析对现有高维多目标鸽群算法存在的算法收敛性和多样性的冲突问题。然后，通过引入基于 KIGD 指标的环境选择策略提高了算法的多样性，通过基于竞争机制的种群更新策略来提高算法的选择压力的同时提高收敛性。最后将设计的算法与其他算法进行相应的仿真实验，并对实验结果进行了相应的分析。

3.1 问题分析

当 MaOPs 中目标个数增加时，传统的 Pareto 支配机制使得算法在迭代后期产生的子代之间几乎都是非支配的。高维多目标鸽群算法在个体选择困难的同时也存在另一个问题，即种群多样性和收敛性冲突加剧的问题。收敛性反应解集与真实 Pareto 前沿之间的逼近程度，多样性反应整个解集在目标空间中的分散程度。收敛性变量是指被扰动的决策变量产生出来的解集值是互相支配的，意味着优化收敛性变量能够使 Pareto 前沿向着理想点的方向移动。多样性变量是指被扰动的决策变量产生出来的解是非支配的，意味着多样性变量能使 Pareto 前沿面向两边扩散。对于函数的一致收敛性定义如下：

定义 3.1：设函数列$\{f_n\}$与函数 f 定义在同一数集 D 上，若对任给的正数 ε，总存在某一正整数 N，使得当 $n>N$ 时，对一切 $x\in D$，都有 $\left|f_n(x)-f(x)\right|<\varepsilon$，则称函数列 $\{f_n\}$ 在 D 上一致收敛与 f。

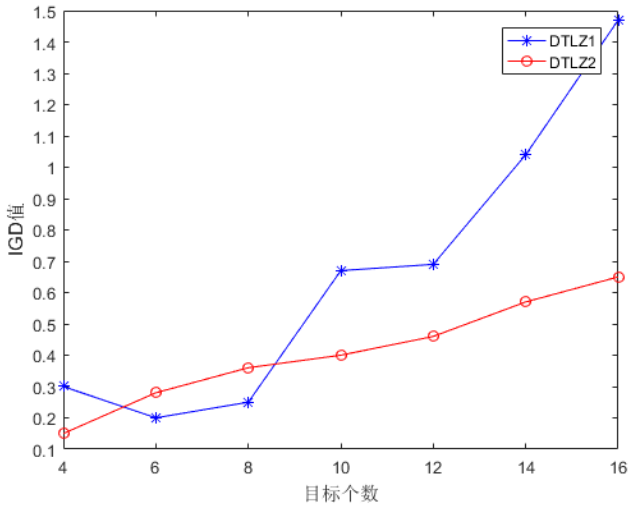


图 3.1 目标个数增加，高维多目标鸽群算法在 DTLZ1 和 DTLZ2 上的 IGD 值

Fig 3.1 IGD values of many-objective pigeon-inspired optimization algorithm on DTLZ1 and DTLZ2 with the number of objectives increases

高维多目标优化算法在解决高维多目标问题时，若单独追求收敛性的提高，基于收敛性原则，会导致算法最终求得的解集可能存在许多相同的解，从而导致算法的多样性变差。若单独追求多样性的提升，此时的解集大多是互相非支配的，这必定导致收敛性变差。图 3.1 列出了高维多目标鸽群算法在 DTLZ1 和 DTLZ2 问题上当目标个数为 4-16 情况下的 IGD 值，IGD[61]是衡量算法综合性能的指标，其值越小代表算法性能越好。由图可以看出，随着目标个数的增加，算法的 IGD 值逐渐加大，因此对应的收敛性和多样性冲突也随之增加。

对于高维多目标优化算法的研究中，众多专家提出了许多衡量算法性能的绩效评价指标，如反转世代距离 IGD[61]，超体积指标 HV[62]，世代距离 GD[63]，改进世代距离 IGD+[64]，$I\varepsilon+$ [65]以及 R2[66]等。利用这些指标衡量算法收敛性、多样性或者综合性能的同时，也有研究者们在算法设计中结合上述指标提高算法的选择效率并在一定程度上提高算法的收敛性和多样性的综合性能。表 3.1 为对常见的基于指标的算法相关研究。

表 3.1 常见的基于指标的算法研究

Table 3.1 Common research on indicator-based algorithms

评价指标	对应算法	原理
超体积指标 HV	HyPE[67], SMA-EMOA[68], MOCMA-ES[69]	通过计算种群中的所有个体与参考点所形成的超体积，衡量算法的综合性能
反转世代距离 IGD	MOEA/IGD-NS[70], AR-MOEA[71]	通过构建类似 PF 的子集，并计算种群中个体与子集的距离来衡量算法综合性能
改进的世代距离 IGD+，IGD$^+$S，$I\varepsilon+$	MaOEA-ITS[72], MOEA/IGD+S[73], IBEA[74], Two_Arch2[75], MOEA/IGD+ S[76]	通过判断解集与真实 Pareto 前沿面的近似程度衡量算法性能
R2	R2-MOEA[77], R2-MOGA[78], MOMBI[79], MOMBI-II[80], R2-MOPSO-II[81],[82]	通过基于理想点与参考点的均匀参考向量，衡量种群收敛性和多样性

如表 3.1 所示，基于 HV 指标的算法在计算中不需要真实 Pareto 解集，具有良好的 Pareto 相容性。基于 IGD 和改进世代距离的算法在一定程度上提高了算法的算法，但对类似 PF 的子集构建存在很大的困难，真实 Pareto 前沿在很多情况下也未知，这使得该类算法对适应度值的求解存在一定的缺陷。基于 R2 指标的算法计算复杂度要低于 HV 指标，但它的计算需要一组均匀分布的参考向量，这也使得算法性能受限于真实 Pareto

前沿。上述算法在一定程度上提高了算法的收敛性能和多样性能，但上述机制均是对种群中的所有个体进行比较与选择，也是对整个种群通过指标性能分析进行进一步的进化操作。而随着所要求解的实际工业问题变得越来越复杂，对应的目标空间的维度越来越高，对整个解集的比较与分析变得越困难，因此考虑种群中的局部感兴趣区域，将基于局部区域性能分析的指标引入高维多目标鸽群算法的环境选择过程中提高算法的性能。所以，本章提出了基于 KIGD 指标的竞争高维多目标鸽群算法。通过基于 KIGD 指标的环境选择策略在保证一定收敛性的同时提高算法的多样性，利用基于竞争机制的种群更新策略来保证算法的收敛性。

3.2 基于 KIGD 指标的竞争高维多目标鸽群算法

在本节中，详细描述了所提出的基于 KIGD 指标的竞争高维多目标鸽群算法（CMAPIO_KIGD）。首先，在 3.2.1 节中详细描述了 CMAPIO_KIGD 的总体框架。主要包括以下两个个方面：基于 KIGD 指标的环境选择策略来提高算法的多样性；基于竞争机制的种群更新策略来提高算法的收敛性。分别在论文中的 3.2.2 和 3.2.3 节进行详细描述。最后，该算法的计算复杂度在 3.2.4 节中进行了相应讨论。

3.2.1 CMAPIO_KIGD 算法主要框架

Algorithm 3.1 CMAPIO_KIGD 的总体框架

开始

初始化种群 P，构建外部归档集 A，均匀分布参考点集 R；

初始化种群中个体速度 V_i，位置 X_i；

计算种群中个体 p_i 的适应度值，局部最优位置 $pbest_i$，局部中心点 P_{center}；

利用现有种群更新外部归档集 A，并计算归档集中个体适应度值;

while $T \le T_{\max}$

根据 3.2.3 节速度位置更新策略判断个体是否需要进行更新

就算更新后种群个体的速度度 V_i，位置 X_i；

计算种群中个体 p_i 的适应度值，更新局部最优位置 $pbest_i$，局部中心点 P_{center}；

$Q=(P,P_{center},P_{best},Archive,R)$；

对父代 Q 执行 3.2.2 节的环境选择策略得到种群 P；

对外部归档集进行更新 A，并对其执行进化策略得到新种群 S；

计算新种群 S 中个体的适应度值;

更新外部归档集 A；

end while

输出：种群 P，外部归档集 A

结束

CMAPIO_KIGD 的主要框架如算法 3.1 所示。同算法 2.1 的过程相似，首先执行初始化操作。种群 P、外部归档集 A 和均匀分布的参考点集 R 被初始化，同时初始化种群中每个鸽子个体的速度 V_i 与位置 X_i。然后，计算现有种群的局部中心点 P_{center} 以及每个鸽子个体 p_i 的目标函数值。将种群中的个体进行非支配排序后将优势个体放到外部归档集 A 中，对现有归档集进行更新。然后执行主要的进化过程。通过在 3.2.3 节中所提的速度位置更新策略判断个体是否需要进行更新，对需要的个体进行更新操作得到新的种群 P 并计算其适应度值，以及种群的局部最优个体 $pbest_i$ 和局部中心个体 P_{center}。然后对由上述个体共同组成的父代种群 Q 进行 3.2.2 节所提的环境选择策略从而得到新的解集性能更好的种群。然后，通过在外部归档集上执行模拟二进制交叉(SBX)和基于多项式的变异(PM)两种进化策略，得到一个新的种群 S。直到循环迭代结束，得到最后的种群 P 和外部归档集 A。

3.2.2 基于 KIGD 指标的环境选择策略

算法 3.2 介绍了基于 KIGD 指标[37]的环境选择策略。首先进行初始化操作，得到种群、参考点集以及参考向量集。然后对种群中的个体进行非支配排序，进而划分层级 $L_1, L_2, \cdots$。计算每一层个体的数量，按照层级的升序排序依次将对应层级的个体放到子代中。当放入种群的某一层个体后子代个数大于 N 时，该层即为关键层。然后对关键层上个体进行 KIGD 排序，选择 k 个个体进入到子代种群中。对应的基于 KIGD 排序选择策略如算法 3.3 所示。

算法 3.2 基于 KIGD 指标的环境选择策略

初始化种群 P，参考点集 R_r，参考向量 V，种群个数 N；
非支配排序 $(Q, R_r) = \{L_1, L_2, \cdots\}$；
for each $L_l \in \{L_1, L_2, \cdots\}$
  if $|P| + |L_l| \le n$ then
    $P = P \cup L_l$
  else
    /*对关键层个体进行 KIGD 指标排序选择*/
    $k = N - |P|$;
    $R_t = KIGDranking(L_l, n - |P|, V, R_r)$;
    $P = P + R_t$;
  end if
end for
输出：种群 P；

KIGD 指标可以衡量拐点区域解集的多样性，分别由拐点区域的均匀分布的参考点集 $\mathbb{R}$ 与所获得的解集 $\mathbb{Q}$ 所组成，KIGD 值越小，算法的多样性能越好，也代表该算法得到的解集可以广泛覆拐点区域。KIGD 的公式如 2.10 所示。其中 $d(v_i,\mathbb{R})$ 为从 $v_i \in \mathbb{R}$ 到近似解集 $\mathbb{Q}$ 中的点的最短欧氏距离，如公式 3.1 所示。它的物理意思为每个参考点到最近的候选解集中个体的平均欧式距离。

$$d(v_i,\mathbb{Q}) = \sqrt{\sum_{i=1}^{m} \max(q_i - v_i, 0)^2} \tag{3.1}$$

算法 3.3 介绍了基于 KIGD 排序选解策略。对于要进行的 KIGD 排序选解策略的种群 P_l，输入对应的参考点 R_l。首先计算每个参考点与种群中所有个体的 KIGD 值。然后将欧式距离最小的个体关联到对应的参考点上。然后通过判断参考点所关联的个体的数目进行接下来的选择操作。其中，当关联的个体数目只有一个时，该个体被选择进入子代；当有多个个体关联到同意参考点时，通过计算这些个体到参考点的垂直距离，距离最小的个体被选择进入子代个体中。循环直到选择足够的个体进入子代。

算法 3.3 基于 KIGD 排序选解策略

输入：种群 P_l，计算 KIGD 所需参考点 R_l；

for $i=1$ to $|R_l|$ do

  for $j=1$ to $|P_l|$ do

    $KIGD_{i,j} = Calculate-KIGD(P_l, R_l)$;

  end for

  $c = \arg\min_{j\in\{1,\ldots,|P_l|\}} KIGD_{i,j}$;

  if $|c| == 1$ then

    $S_i = c$;

  else if $|c| > 1$ then

    for $n=1$ to $|c|$ do

      /*计算个体到所关联的参考向量的距离*/

      $D = ver-dis(P_l, R_l)$;

      $S_i = c(D)$;

    end for

  end if

end for

$P_l^{KIGD} = P_l(S)$;

输出：通过 KIGD 指标选择的个体 P_l^{KIGD}；

3.2.3 基于竞争机制的种群更新策略

本节介绍了一种新的速度位置更新策略，通过引入竞争机制[83, 84]提高算法的收敛性能。如算法 3.4 所示，首先初始化种群 $P*$，并根据公式 3.2 计算种群 P 中每个鸽群个体的适应度值。然后从种群中任意选择两个个体 $\{p,q\}$，判断他们对应的适应度函数值。适应度值小的作为胜利者采用新的速度位置更新策略进入下一代，而适应度值相对较大的作为需要学习的个体利用胜利者的信息进行相应的速度位置更新。接着，对新产生的种群 $P*$ 也采用多样式变异进化策略来防止算法陷入局部最优。整个个体更新策略一直重复到整个种群被遍历。

算法 3.4 基于竞争机制的个体更新策略

开始

输入：种群 P；

初始化种群 $P*$；

计算每个鸽群个体的适应度值；

while $|P|>1$ do

随机从种群中选择两个个体 $\{p,q\}$；

$P \leftarrow P-\{p,q\}$；

if $Fitness(p)<Fitness(q)$ then

$X_w \leftarrow p$； /*胜利者*/

$X_l \leftarrow q$； /*Loser 个体需要学习胜利者经验*/

else

$X_w \leftarrow q$； /*胜利者*/

$X_l \leftarrow p$； /*Loser 个体需要学习胜利者经验*/

end if

X_l 通过学习 X_w 进行更新；

对 $\{X_l, X_w\}$ 执行进化操作；

$P* \leftarrow P* \cup \{X_l, X_w\}$；

end while

输出：新种群 $P*$；

结束

$$Fitness(p)=\min_{q\in P且q\neq p}\sqrt{\sum_{i=1}^{m}\left(\max\left\{0, f_i\left(\vec{q}\right)-f_i\left(\vec{p}\right)\right\}\right)^2} \tag{3.2}$$

其中，$f_i\left(\vec{p}\right)$ 是个体 p 在第 i 个目标上的目标函数值，M 是目标空间的维度。本文利用 SDE 策略衡量个体在种群中的收敛性和多样性。

对于种群中的个体更新主要分为两种，对于竞争胜利者采用如下的速度更新公式如下所示：

$$V_i\left(t_{now}+1\right)=e^{-Rt_{now}}\cdot V_i\left(t_{now}\right)+r_1\cdot tr\cdot\left(1-\log_T^{t_{now}}\right)\cdot\left(X_{glo}-X_i\left(t_{now}\right)\right)+ r_2\cdot tr\cdot\log_T^{t_{now}}\cdot\left(X_{cen}-X_i\left(t_{now}\right)\right)+r_3\cdot\left(X_{glo}-X_{cen}\right) \tag{3.3}$$

$$X_i\left(t_{now}+1\right)=X_i\left(t_{now}\right)+V_i\left(t_{now}\right) \tag{3.4}$$

其中t_{now}表示当前迭代次数，T代表最大迭代次数，R为地图与指南针算子，tr是确保从地图与指南针算子平稳过渡到地标算子之间的迁移因子。X_{glo}为个体全局最优位置，X_{cen}为当前迭代中某些个体的中心位置信息，可根据公式(3.5)计算。

$$X_{cen}=\frac{\sum_{j=1}^{n_1^x}S_{1j}^X}{n_1^x}S_1^X n_1^X \tag{3.5}$$

其中$\sum_{}^{n_1^x}S_{1j}^X$表示非支配集合中个体的所有解的和S_1^X，n_1^X是集合S_1^X中解的个数。r_1，r_2和r_3表示三个学习因子，定义如第二章公式 2.7。

对于竞争失败的鸽群个体，通过学习胜利者信息进行速度位置更新。具体更新公式如下所示。其中，c_0和c_1为属于$[0,1]$之间均匀分布的随机数。

$$V_i\left(t_{now}+1\right)=c_0\cdot V_i\left(t_{now}\right)+c_1\cdot\left(X_w-X_i\left(t_{now}\right)\right) \tag{3.6}$$

$$X_i\left(t_{now}+1\right)=X_i\left(t_{now}\right)+V_i\left(t_{now}+1\right)+c_0\left(V_i\left(t_{now}+1\right)-V_i\left(t_{now}\right)\right) \tag{3.7}$$

3.2.4 CMAPIO_KIGD 时间复杂度分析

本节主要介绍 CMAPIO_KIGD 算法的时间复杂度。CMAPIO_KIGD 算法的时间复杂度主要来自于基于 KIGD 指标的环境选择策略、基于竞争机制的更新策略和外部归档集更新机制。其中基于 KIGD 指标的环境选择中主要包括 Pareto 非支配排序选择和基于 KIGD 指标选解策略。当种群大小为N，目标空间维度为M时，非支配排序的计算复杂度为$O\left(N^2M\right)$。基于 KIGD 指标选解策略需要的$O\left(N^2M\right)$的计算开销。采用竞争机制的种群更新策略中，每次竞争中需要选择 2 个鸽群个体进行适应度比较，因此相应的时间复杂度为$O\left(2N\times\log\left(2N\right)\right)$，比较之后的个体的速度位置更新需要$O\left(N\right)$的时间复杂度。归档集更新的计算复杂度主要是通过将种群中的个体与归档集中的精英个体进行比较得到的。因为归档集的大小通常是根据种群的大小来设定的，因此最差情况下它的时间复杂度为$O\left(N^2M\right)$。综上所述，当处于最差情况下，CMAPIO_KIGD 算法的整体时间复杂度为$O\left(N^2M\right)$。

3.3 仿真实验

为了验证本章所提的 CMAPIO_KIGD 算法的实验性能，本部分将所提的 CMAPIO_KIGD 算法与与 NSGA-III[8]，GrEA[9]，MOEA/D[54]，RVEA[55]和 MAPIO[35]在标准测试集 DTLZ[38]和 WFG[39]上的性能进行了分析，并利用 IGD 和 HV 对其算法收敛性和多样性等综合性能进行衡量。

3.3.1 参数设置及评价指标

测试集以及相关参数介绍如第二章 2.3.1 与 2.3.2 所示。本章利用新的反转世代距离 IGD 和超体积指标 HV 对算法在 DTLZ 和 WFG 测试集上的收敛性与多样性等综合性能进行衡量。下面对相应的评价指标进行简要介绍。

（1）评价指标

反转世代距离 IGD[61]表示从每个参考点到最近的解的欧式距离的平均值。IGD 值越小，算法的综合性能越好。IGD 值可由式 3.8 计算:

$$IGD\left(Q,Q^*\right)=\frac{\sum_{x\in Q^*}\min_{y\in Q} dis\left(x,y\right)}{\left|Q^*\right|} \tag{3.8}$$

其中 Q 是由算法得到的近似解集。Q^* 表示在真实 Pareto 前沿中均匀分布并采样的参考点集。$dis\left(x,y\right)$ 代表参考点集合 Q^* 中的点 x 到近似解集 Q 中点 y 的欧式距离。IGD 值的计算，不仅计算效率高，而且能衡量算法的收敛性和多样性。

超体积指标 HV[62]是算法求得的 Pareto 解集与算法所得的参考点在目标空间中所组成区域的超体积。HV 值越高，算法的综合性能越好。HV 的计算如 3.9 所示。其中 δ 表示体积的勒贝格测度。$|s|$ 是 Pareto 前沿上非支配解的个数。v_i 是 Pareto 解集中第 i 个个体与参考点形成的超体积值。

$$HV=\delta\left(\bigcup_{i=1}^{|S|}v_i\right) \tag{3.9}$$

3.3.2 在 DTLZ 标准测试集上的实验结果及分析

本节为了验证所提的 CMAPIO_KIGD 算法的性能，将所提的将所提出的算法与其他五种多目标优化算法在 DTLZ 测试函数上进行了比较，并通过 HV 指标对算法结果进行了评价。表 3.2 显示了所提的 CMAPIO_KIGD 与 GrEA、NSGA-III、RVEA、MOEA/D 和 MAPIO 五种经典算法在 DTLZ 测试函数上 IGD 指标的比较结果。其中，表格中加粗黑体显示的值表示算法求得的最优解，“+”、“-”和“=”分别代表其他算法性能优于、劣于或者等于所提的 CMAPIO_KIGD 算法。从表 3.2 最后一行可以看出，在 35 个测试函数中，CMAPIO_KIGD 与 NSGA-III 相比具有明显的优势，NSGA-III 在任何函数上的表现

都不比其他函数好。在 2 个测试问题中比所提算法好，但在 25 个问题中性能比我们所提算法差。GrEA 和 MOEA/D 与本文算法相比性能相近。在十多个测试问题中性能要比所提的算法要好，但是在近 20 个问题中算法求得的性能不如 CMAPIO_KIGD。MOEA/D 算法在 DTLZ1 测试函数上的性能优于其他算法，这是因为测试函数具有多模态的特点，且难以获得收敛解，基于分解的 MOEA/D 算法在求解这类问题上具有较好的性能。在与 RVEA 和 MAPIO 的比较结果中，本文算法有相对较好的优势，在 11 个测试问题上比上述算法差，但在 16 个测试问题中比它们要好。突出显示区域显示了 CMAPIO_KIGD 算法在 DTLZ3 和 DTLZ7 上的优越性能。

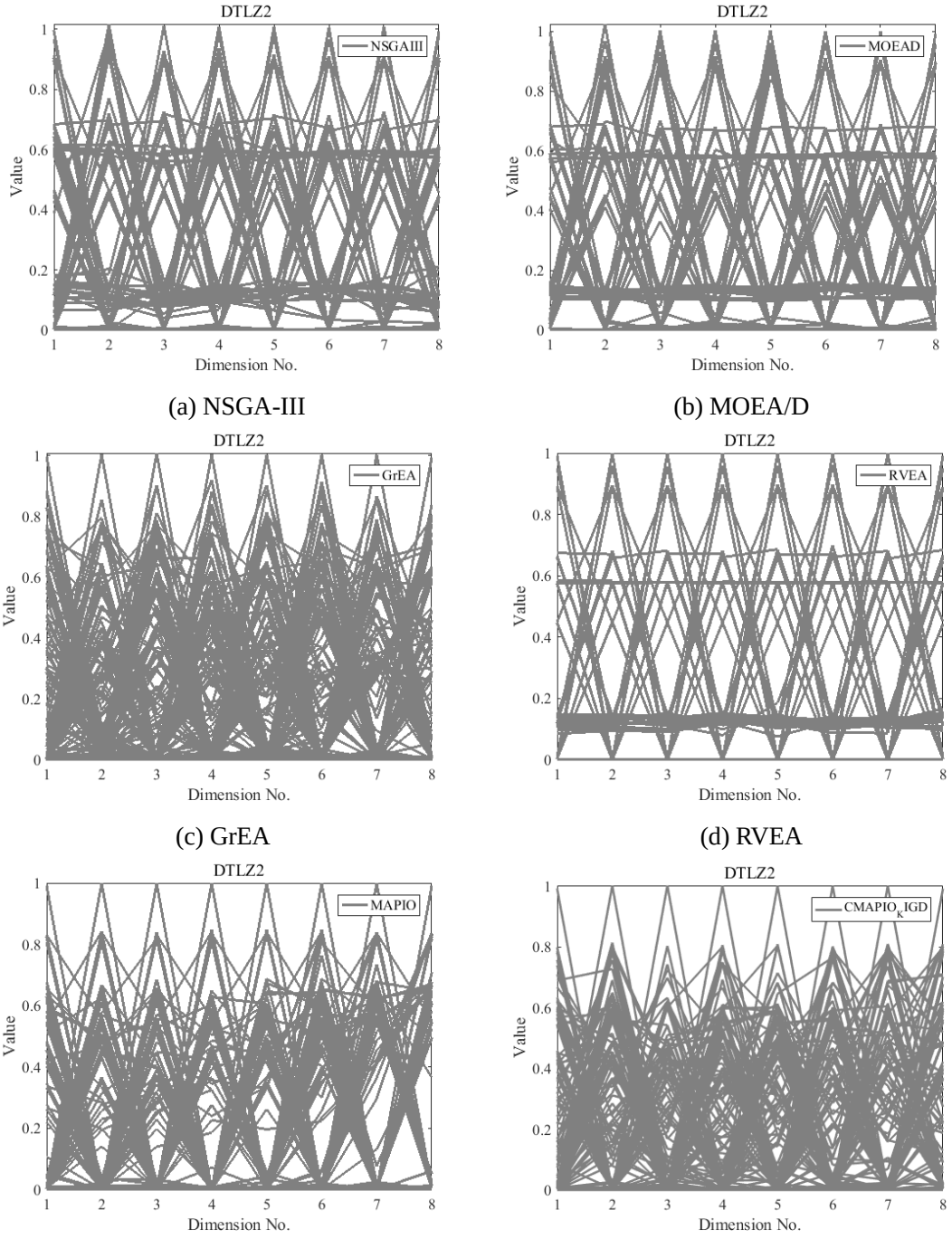
表 3.2 六种算法在 DTLZ 测试问题上不同目标的 HV 值

Table 3.2 The HV values of these algorithms for different objectives in the DTLZ test problems

Problem	M	D	NSGAIII	GrEA	MOEAD	RVEA	MAPIO	CMAPIO_KIGD
DTLZ1	4	8	2.4987e-1 (3.15e-1) -	4.3383e-1 (3.67e-1) =	5.6713e-1 (3.79e-1) =	8.6552e-2 (1.87e-1) -	**6.0588e-1 (3.64e-1) =**	5.9330e-1 (1.95e-1)
	6	10	8.1696e-3 (2.81e-2) -	6.5264e-2 (2.08e-1) -	**5.7604e-1 (3.27e-1) +**	1.6731e-1 (3.00e-1) -	5.1964e-1 (3.16e-1) +	2.4100e-1 (1.36e-1)
	8	12	0.0000e+0 (0.00e+0) -	2.5136e-2 (9.74e-2) -	**6.4355e-1 (1.85e-1) +**	1.4608e-1 (2.46e-1) =	4.5773e-1 (4.25e-1) =	2.0132e-1 (1.56e-1)
	10	14	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	**5.5328e-1 (2.38e-1) +**	4.4173e-2 (7.77e-2) -	1.1114e-1 (2.48e-1) -	2.1378e-1 (1.43e-1)
	15	19	1.6624e-3 (3.57e-3) -	1.1296e-4 (4.37e-4) -	3.2212e-1 (1.92e-1) =	2.1322e-1 (2.85e-1) =	3.2073e-1 (3.13e-1) -	**3.2636e-1 (1.58e-1)**
DTLZ2	4	13	6.8748e-1 (2.82e-3) =	**7.0030e-1 (2.01e-3) +**	6.9127e-1 (2.49e-3) +	6.8873e-1 (3.44e-3) +	6.0147e-1 (1.86e-3) -	6.8241e-1 (1.13e-2)
	6	15	7.9817e-1 (1.02e-2) +	**8.3432e-1 (4.08e-3) +**	8.2938e-1 (8.07e-3) +	8.3300e-1 (3.43e-3) +	7.5481e-1 (1.75e-3) =	7.7629e-1 (1.75e-2)
	8	17	8.1977e-1 (5.28e-2) -	7.1789e-1 (3.77e-2) -	8.5604e-1 (1.42e-2) -	8.9172e-1 (8.11e-3) -	9.2370e-1 (1.98e-3) -	**9.5447e-1 (5.52e-2)**
	10	19	6.4772e-1 (8.17e-2) =	8.9061e-1 (1.09e-2) +	8.4887e-1 (2.88e-2) +	9.0243e-1 (1.16e-2) +	**9.4098e-1 (1.03e-2) +**	6.8727e-1 (3.70e-2)
	15	24	5.9916e-1 (1.26e-1) -	9.0921e-1 (1.79e-2) +	8.4452e-1 (7.49e-2) +	**9.4734e-1 (1.07e-2) +**	9.2210e-1 (2.69e-2) +	7.2191e-1 (4.39e-2)
DTLZ3	4	13	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	1.0757e-2 (4.17e-2) -	**1.4249e-1 (8.20e-2)**
	6	15	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	**5.3963e-2 (1.45e-2)**
	8	17	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	3.1919e-3 (1.24e-2) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	**2.2046e-2 (2.74e-2)**
	10	19	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	**2.4628e-2 (2.39e-2)**

	15	24	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	**1.6297e-2 (2.23e-2)**
DTLZ4	4	13	6.6791e-1 (5.15e-2) -	6.1176e-1 (1.12e-1) -	4.4427e-1 (1.60e-1) -	6.8930e-1 (3.03e-3) =	6.8510e-1 (3.65e-2) -	**6.9324e-1 (2.34e-2)**
	6	15	7.6976e-1 (5.08e-2) =	8.3899e-1 (1.45e-2) +	6.3242e-1 (2.05e-1) -	8.2512e-1 (2.59e-2) +	**8.4377e-1 (2.50e-2) +**	7.9625e-1 (3.78e-2)
	8	17	7.7534e-1 (9.98e-2) =	8.6872e-1 (9.94e-3) =	5.5327e-1 (1.97e-1) -	**9.0785e-1 (4.45e-3) +**	8.9905e-1 (2.68e-2) +	8.4477e-1 (5.80e-2)
	10	19	7.1626e-1 (1.47e-1) -	9.4886e-1 (4.47e-3) +	7.3938e-1 (1.62e-1) =	9.3879e-1 (6.63e-3) +	**9.6648e-1 (5.47e-3) +**	8.1124e-1 (4.48e-2)
	15	24	5.2456e-1 (1.95e-1) -	9.6283e-1 (9.42e-3) +	5.4150e-1 (2.53e-1) -	**9.8686e-1 (1.83e-3) +**	9.7557e-1 (1.13e-2) -	9.7748e-1 (2.24e-2)
DTLZ5	4	13	1.3299e-1 (5.83e-3) -	1.0815e-1 (9.91e-3) -	1.4219e-1 (5.78e-3) -	1.0436e-1 (8.21e-3) -	1.4684e-1 (1.03e-3) =	**1.4694e-1 (2.52e-3)**
	6	15	1.7835e-2 (2.65e-2) -	4.0316e-2 (2.40e-2) -	**1.1400e-1 (3.66e-3) +**	8.3881e-2 (7.23e-3) -	9.4523e-2 (1.25e-2) =	9.8472e-2 (1.56e-2)
	8	17	3.0468e-2 (2.26e-2) -	1.0438e-2 (1.65e-2) -	1.0167e-2 (7.39e-3) -	**8.4322e-2 (7.24e-3) =**	4.4816e-2 (2.48e-2) -	6.6378e-2 (2.86e-2)
	10	19	1.2473e-2 (1.69e-2) -	1.4574e-4 (2.15e-4) -	1.0012e-2 (2.86e-4) -	**6.0826e-2 (2.20e-2) =**	1.2795e-2 (1.86e-2) -	5.3035e-2 (2.70e-2)
	15	24	1.2193e-2 (1.84e-2) -	1.2243e-3 (4.74e-3) -	3.4433e-2 (3.07e-4) -	**9.0720e-2 (7.01e-4) +**	1.7419e-2 (2.81e-2) -	5.0478e-2 (3.46e-2)
DTLZ6	4	13	1.7674e-2 (3.62e-2) +	9.9364e-2 (2.81e-2) +	9.2965e-2 (5.65e-2) +	4.3035e-2 (4.76e-2) +	**1.2480e-1 (1.18e-2) +**	1.7938e-3 (5.10e-3)
	6	15	0.0000e+0 (0.00e+0) =	2.0102e-2 (3.48e-2) +	5.1676e-2 (4.85e-2) +	2.4672e-2 (3.58e-2) +	**7.6863e-2 (3.34e-2) +**	8.2790e-7 (2.18e-6)
	8	17	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	**4.0468e-2 (4.89e-2) =**	1.8880e-2 (3.89e-2) =	1.0199e-2 (2.48e-2) =	3.5297e-5 (1.37e-4)
	10	19	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**7.3147e-2 (4.25e-2) +**	7.9835e-7 (3.09e-6) =	1.4594e-2 (2.86e-2) +	2.8273e-11 (1.10e-10)
	15	24	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**8.8308e-2 (2.40e-2) +**	8.2508e-3 (2.44e-2) =	0.0000e+0 (0.00e+0) =	3.5111e-16 (9.27e-16)
DTLZ7	4	23	1.7591e-1 (1.55e-2) =	**2.5182e-1 (3.89e-3) +**	7.7522e-2 (3.31e-2) -	1.6122e-1 (2.20e-2) -	2.1291e-1 (2.13e-2) +	1.8422e-1 (8.09e-3)
	6	25	4.6760e-2 (2.15e-2) -	**1.9753e-1 (1.26e-2) +**	5.5480e-3 (6.32e-3) -	6.0811e-2 (2.65e-2) -	1.8972e-1 (1.33e-2) +	1.4776e-1 (2.14e-2)
	8	27	4.1204e-4 (3.68e-4) -	1.1902e-1 (1.52e-2) =	1.5906e-3 (2.85e-3) -	8.5652e-3 (1.15e-2) -	1.1631e-1 (5.77e-2) =	**1.2856e-1 (1.44e-2)**
	10	29	0.0000e+0 (0.00e+0) -	1.2833e-3 (2.48e-3) -	4.3639e-5 (8.83e-5) -	5.7808e-4 (1.27e-3) -	5.6912e-2 (8.18e-2) -	**1.1519e-1 (9.56e-3)**
	15	34	4.9492e-6 (1.91e-5) -	1.4263e-4 (5.52e-4) -	3.0682e-5 (1.16e-4) -	2.3282e-4 (8.06e-4) -	1.0849e-1 (6.16e-2) -	**1.1041e-1 (9.64e-3)**
+/-/=			2/25/8	10/20/5	12/19/4	11/16/8	11/16/8	

为进一步分析上述算法求得的解集在目标空间中的分布情况，图 3.2 对 DTLZ2 测试集上 8 个目标情况下上述算法所得的 Pareto 解集进行介绍。由图可知，MOEA/D、NSGA-III、RVEA 和 MAPIO 算法得到的 Pareto 前沿分布相对较差。GrEA 算法得到的分布比较良好，说明了在多样性和收敛性方面的综合性能。CMAPIO_KIGD 算法的分布性与 GrEA 算法的性能相近。



(a) NSGA-III　(b) MOEA/D

(c) GrEA　(d) RVEA

(e) MAPIO　(f) CMAPIO_KIGD

图 3.2 不同算法当目标个数为 8 时在 DTLZ2 测试函数上的 Pareto 前沿

Fig 3.2 Pareto Front obtained by different algorithms on the 8-objective DTLZ test function

3.3.3 在 WFG 标准测试集上的实验结果分析

表 3.3 进一步展示了 CMAPIO_KIGD 与 NSGA-III、GrEA、MOEA/D、RVEA、MAPIO 在 WFG 测试函数上使用 IGD 指标的比较实验结果。从表 3.3 的最后一行可以看出，与 NSGA-III 相比，NSGA-III 在 18 个问题上性能好，但在 22 个问题上性能比所提算法差。这是因为使用 Pareto 支配关系和参考点策略在目标个数急剧增加时缺乏选择压力，从而导致较差的收敛性能。与 MOEA/D 算法相比，本文所提算法具有显著性优势，在 38 个测试问题中性能更好一点。在与 GrEA 和 RVEA 的比较结果中，GrEA 在 21 个测试问题中略差于所提的 CMAPIO_KIGD 算法，RVEA 在 22 个问题上略差于本文算法。在与 MAPIO 的比较结果中，本文提出的算法也具有明显的优势，在 28 个测试问题上要优于 MAPIO 算法。虽然本文算法在 WFG1 上的性能没有超过其他算法，但在整个测试函数集上，尤其是在 WFG3、WFG4 和 WFG9 测试函数上的性能优于其他算法。CMAPIO_KIGD 中基于 KIGD 指标的选择策略能够在提高算法在局部拐点区域的多样性能力。基于 KIGD 指标的选解策略能够保证算法在局部区域中的多样性。而所提算法在求解目标个数为 10 和 15 时，在多个测试集上性能要强于其他算法，也证明了所提算法在解决更高维度问题时的卓越性能。

表 3.3 六种算法在 WFG 测试问题上不同目标的 IGD 值

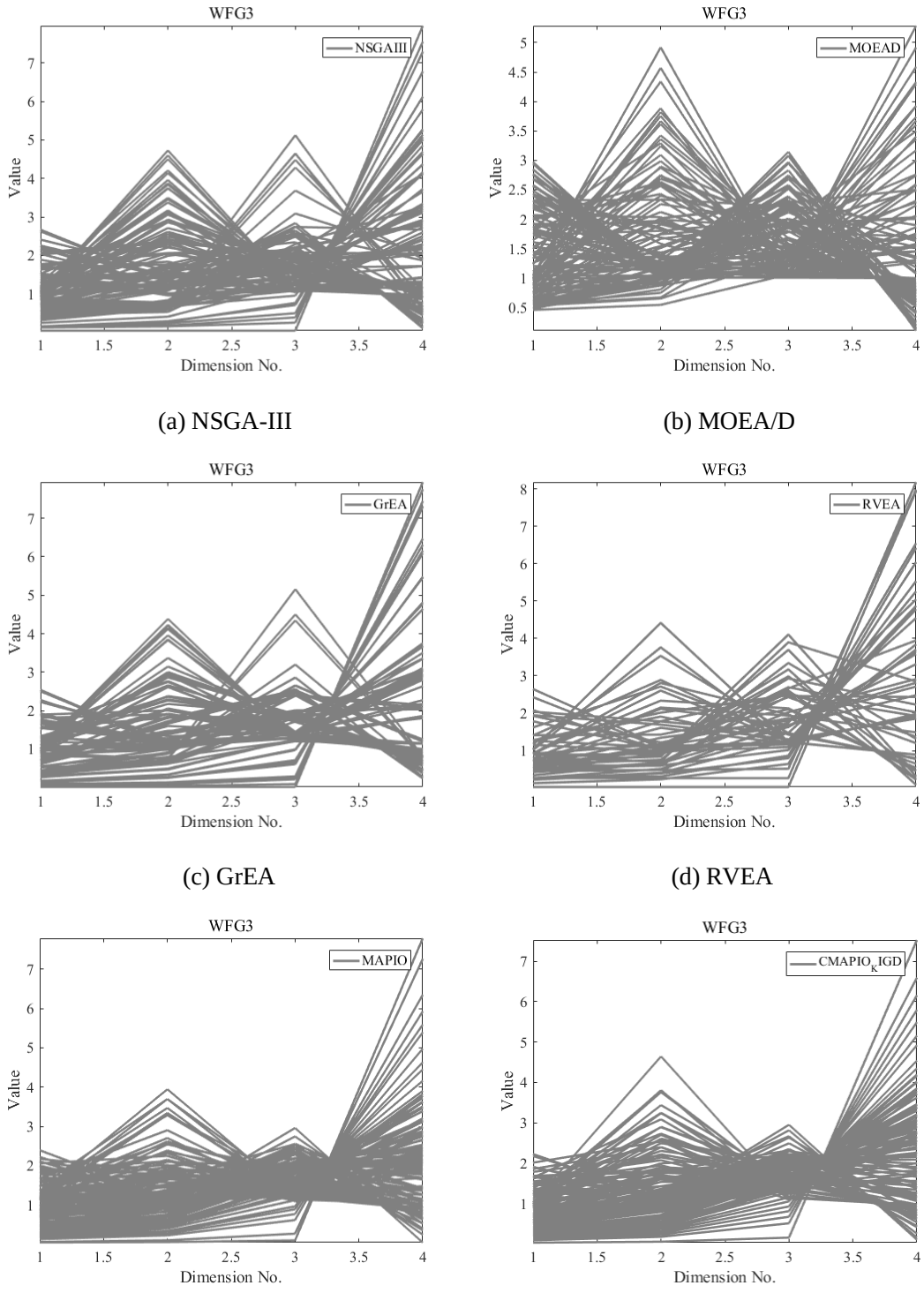
Table 3.3 The IGD values of these algorithms for different objectives in the WFG test problems

Problem	M	D	NSGAIII	GrEA	MOEAD	RVEA	MAPIO	CMAPIO_KIGD
WFG1	4	13	1.5518e+0 (1.16e-1) -	1.2905e+0 (8.29e-2) -	1.2916e+0 (1.27e-1) -	1.4477e+0 (9.90e-2) -	1.4104e+0 (1.06e-1) -	**1.2671e+0 (1.57e-1)**
	6	15	2.0901e+0 (8.86e-2) =	2.2836e+0 (1.50e-1) -	**1.6287e+0 (1.30e-1) +**	1.9057e+0 (1.33e-1) +	1.8756e+0 (2.83e-1) +	2.1601e+0 (1.95e-1)
	8	17	2.6203e+0 (6.97e-2) =	2.7812e+0 (1.06e-1) -	**2.1583e+0 (1.56e-1) +**	2.4009e+0 (9.30e-2) +	2.5350e+0 (2.61e-1) =	2.6235e+0 (1.50e-1)
	10	19	3.1301e+0 (5.17e-2) -	2.9242e+0 (9.29e-2) =	**2.6303e+0 (1.89e-1) +**	2.8686e+0 (4.10e-2) +	2.9163e+0 (1.82e-1) +	3.0605e+0 (7.83e-2)
	15	24	3.8450e+0 (1.18e-1) =	3.3825e+0 (2.38e-1) +	3.8656e+0 (9.78e-2) =	3.5159e+0 (1.78e-1) +	**3.3532e+0 (2.12e-1) +**	3.8763e+0 (1.92e-1)
WFG2	4	13	**3.1840e-1 (8.86e-3) +**	3.9329e-1 (1.52e-2) +	6.0453e-1 (1.08e-1) =	3.3052e-1 (8.69e-3) +	6.3609e-1 (9.54e-2) =	5.8138e-1 (6.01e-2)
	6	15	**6.3639e-1 (2.41e-2) +**	6.4244e-1 (1.42e-2) =	1.2128e+0 (6.28e-2) -	6.3815e-1 (3.78e-2) +	9.3665e-1 (8.51e-2) -	6.4478e-1 (7.37e-2)
	8	17	1.0594e+0 (1.38e-1) +	1.1414e+0 (5.10e-2) -	1.9168e+0 (1.18e-1) -	**1.0403e+0 (5.96e-2) +**	1.3265e+0 (1.20e-1) -	1.1228e+0 (9.52e-2)
	10	19	1.1085e+0 (3.09e-2) -	1.2089e+0 (4.49e-2) -	2.0675e+0 (6.30e-2) -	1.0821e+0 (4.43e-2) -	1.2233e+0 (5.93e-2) -	**1.0244e+0 (1.37e-1)**

	15	24	1.9984e+0 (1.34e-1) +	1.9990e+0 (9.06e-2) +	3.0205e+0 (1.03e+0) -	**1.8206e+0 (2.23e-1) +**	2.2451e+0 (1.50e-1) =	2.4605e+0 (3.17e-1)
WFG3	4	13	3.6760e-1 (3.83e-2) =	2.7881e-1 (3.77e-2) =	7.6089e-1 (2.18e-1) -	4.1933e-1 (5.03e-2) =	4.5796e-1 (7.23e-2) -	**4.1710e-1 (2.88e-1)**
	6	15	9.9488e-1 (1.59e-1) -	7.6777e-1 (8.64e-2) =	1.9050e+0 (4.79e-1) -	1.1612e+0 (4.20e-1) -	8.5191e-1 (2.19e-1) -	**7.7047e-1 (3.93e-1)**
	8	17	1.0249e+0 (2.05e-1) =	1.1818e+0 (1.64e-1) -	3.9880e+0 (2.76e-1) -	1.8534e+0 (2.35e-1) -	1.1875e+0 (9.19e-1) -	**1.0474e+0 (5.37e-1)**
	10	19	1.2624e+0 (2.21e-1)-	1.7785e+0 (1.51e-1) =	7.0569e+0 (1.08e+0) -	2.3225e+0 (2.70e-1) -	1.4278e+0 (5.71e-1) -	**1.3309e+0 (7.00e-1)**
	15	24	4.7936e+0 (2.19e+0) -	1.6715e+0 (6.00e-1) -	1.2841e+1 (1.56e+0) -	6.0546e+0 (6.66e-1) -	9.3734e-1 (5.32e-1) -	**8.6488e-1 (1.39e+0)**
WFG4	4	13	**6.1291e-1 (1.72e-3) +**	6.3740e-1 (9.66e-3) +	6.9058e-1 (4.96e-2) =	6.1374e-1 (3.34e-3) +	7.1877e-1 (2.14e-2) -	6.9717e-1 (2.47e-2)
	6	15	1.7292e+0 (8.97e-3) -	1.6779e+0 (9.83e-3) -	4.1476e+0 (1.47e-1) -	1.7222e+0 (8.66e-3) -	1.8231e+0 (3.85e-2) -	**1.5718e+0 (2.06e-2)**
	8	17	3.0409e+0 (8.74e-2) =	3.1098e+0 (1.55e-2) -	7.3078e+0 (1.40e-1) -	3.1875e+0 (8.42e-2) -	3.0196e+0 (3.17e-2) =	**3.0145e+0 (3.53e-2)**
	10	19	4.3673e+0 (1.31e-1) -	4.2941e+0 (1.76e-1) -	9.7027e+0 (1.07e-1) -	4.3689e+0 (7.42e-2) -	**4.0347e+0 (2.87e-2) +**	4.1428e+0 (1.10e-1)
	15	24	9.3316e+0 (3.87e-1) -	1.0959e+1 (5.81e-1) -	1.6413e+1 (2.35e+0) -	9.6766e+0 (9.61e-1) -	1.0178e+1 (8.99e-1) -	**8.5028e+0 (1.40e-1)**
WFG5	4	13	**6.0306e-1 (5.19e-3) +**	6.3213e-1 (1.18e-2) +	7.9625e-1 (6.61e-2) -	6.0498e-1 (3.14e-3) +	7.2503e-1 (1.89e-2) -	6.7785e-1 (2.16e-2)
	6	15	1.6927e+0 (1.36e-2) =	1.6628e+0 (1.36e-2) =	4.0096e+0 (8.30e-2) -	1.7115e+0 (9.46e-3) -	1.8640e+0 (3.66e-2) -	**1.6567e+0 (3.91e-2)**
	8	17	3.0236e+0 (3.24e-2) =	3.1343e+0 (1.93e-2) -	7.0838e+0 (1.00e-1) -	3.2583e+0 (5.65e-2) -	3.1108e+0 (4.13e-2) -	**3.0453e+0 (4.44e-2)**
	10	19	4.3405e+0 (8.23e-2) -	4.1189e+0 (3.72e-2) =	9.5070e+0 (8.77e-2) -	4.2358e+0 (7.46e-2) =	**4.0699e+0 (3.85e-2) =**	4.1993e+0 (1.74e-1)
	15	24	8.9175e+0 (1.70e-1) -	1.0204e+1 (6.13e-1) -	1.6344e+1 (2.03e-1) -	8.7013e+0 (3.29e-1) -	8.6472e+0 (1.31e-1) -	**8.3762e+0 (1.80e-1)**
WFG6	4	13	**6.3944e-1 (1.44e-2) +**	6.7282e-1 (1.74e-2) +	8.7769e-1 (1.12e-1) -	6.5187e-1 (1.43e-2) +	7.4504e-1 (1.74e-2) -	6.8897e-1 (1.87e-2)
	6	15	1.7389e+0 (1.11e-2) +	**1.7162e+0 (2.10e-2) +**	4.6833e+0 (2.00e-1) -	1.7576e+0 (3.89e-2) +	1.9368e+0 (3.57e-2) -	1.8058e+0 (3.93e-2)
	8	17	3.0745e+0 (3.54e-2) +	**2.9925e+0 (2.19e-2) +**	7.6905e+0 (1.50e-1) -	3.3860e+0 (9.58e-2) -	3.2434e+0 (6.48e-2) -	3.1413e+0 (5.96e-2)
	10	19	4.4110e+0 (7.83e-2) -	4.1775e+0 (6.43e-2) -	9.9721e+0 (1.49e-1) -	4.1852e+0 (4.46e-2) -	4.1382e+0 (5.11e-2) -	**4.0208e+0 (1.28e-1)**
	15	24	9.2870e+0 (4.39e-1) -	9.3418e+0 (4.78e-1) -	1.6497e+1 (1.98e-1) -	8.6798e+0 (3.87e-1) =	8.7727e+0 (1.46e-1) -	**8.5983e+0 (2.50e-1)**
WF	4	1	**6.2690e-1**	6.6200e-1	8.9402e-1	6.3178e-1	7.4925e-1	6.8864e-1

G7		3	**(7.77e-3) +**	(1.11e-2) +	(1.51e-1) -	(1.12e-2) +	(1.94e-2) -	(1.73e-2)
	6	1	1.7232e+0	**1.6858e+0**	4.2802e+0	1.7359e+0	1.8948e+0	1.7836e+0
		5	(9.86e-3) +	**(1.18e-2) +**	(2.14e-1) -	(1.24e-2) +	(2.47e-2) -	(2.63e-2)
	8	1	3.1107e+0	**2.9316e+0**	7.4892e+0	3.1384e+0	3.1181e+0	3.0643e+0
		7	(1.28e-1) =	**(1.94e-2) +**	(1.47e-1) -	(5.33e-2) -	(4.39e-2) -	(5.59e-2)
	1	1	4.4789e+0	4.2454e+0	9.8617e+0	4.3652e+0	4.0736e+0	**4.0112e+0**
	0	9	(7.44e-2) -	(6.54e-2) -	(2.00e-1) -	(8.28e-2) -	(3.43e-2) =	**(3.13e-1)**
	1	2	1.0013e+1	9.6124e+0	1.7161e+1	9.4819e+0	9.3895e+0	**9.1556e+0**
	5	4	(5.31e-1) -	(4.36e-1) -	(2.71e-1) -	(9.67e-1) =	(5.90e-1) =	**(5.13e-1)**
WFG8	4	1	**6.8084e-1**	6.8489e-1	7.6460e-1	6.8785e-1	7.4702e-1	8.7951e-1
		3	**(5.66e-3) +**	(1.31e-2) +	(7.93e-2) +	(6.41e-3) +	(2.82e-2) +	(6.63e-2)
	6	1	1.8342e+0	**1.7974e+0**	3.8504e+0	1.8220e+0	1.8980e+0	2.0573e+0
		5	(1.00e-1) +	**(1.29e-2) +**	(3.74e-1) -	(2.89e-2) +	(3.63e-2) +	(7.60e-2)
	8	1	3.4843e+0	**3.2700e+0**	6.6055e+0	3.3255e+0	3.3113e+0	3.4216e+0
		7	(2.37e-1) =	**(2.38e-2) +**	(3.73e-1) -	(5.46e-2) +	(3.86e-2) +	(1.14e-1)
	1	1	4.9372e+0	5.3624e+0	8.8565e+0	4.7890e+0	4.4653e+0	**4.4204e+0**
	0	9	(1.43e-1) -	(2.87e-1) -	(4.85e-1) -	(1.43e-1) -	(3.56e-2) =	**(2.08e-1)**
	1	2	1.0466e+1	1.0850e+1	1.3805e+1	9.4807e+0	9.1089e+0	**8.5188e+0**
	5	4	(8.45e-1) -	(2.31e-1) -	(2.99e+0) -	(9.25e-1) -	(5.94e-1) -	**(2.78e-1)**
WFG9	4	1	6.0505e-1	**6.0291e-1**	8.0199e-1	6.2211e-1	6.6667e-1	6.4695e-1
		3	(1.36e-2) +	**(8.88e-3) +**	(5.31e-2) -	(2.16e-2) +	(1.39e-2) -	(1.40e-2)
	6	1	1.6978e+0	1.6433e+0	3.8959e+0	1.7004e+0	1.7523e+0	**1.6398e+0**
		5	(2.35e-2) =	(8.73e-3) =	(2.72e-1) -	(1.80e-2) =	(2.31e-2) -	**(2.01e-2)**
	8	1	3.1489e+0	2.9803e+0	7.0005e+0	3.1324e+0	2.9020e+0	**2.8993e+0**
		7	(8.89e-2) -	(3.78e-2) -	(2.30e-1) -	(5.71e-2) -	(2.41e-2) =	**(3.71e-2)**
	1	1	4.5021e+0	4.1576e+0	9.4300e+0	4.2537e+0	4.0845e+0	**4.0373e+0**
	0	9	(1.58e-1) -	(3.38e-2) =	(3.48e-1) -	(1.02e-1) -	(2.75e-2) =	**(1.15e-1)**
	1	2	9.2951e+0	9.9959e+0	1.5122e+1	8.4879e+0	8.4018e+0	**8.3512e+0**
	5	4	(4.02e-1) -	(4.00e-1) -	(1.83e+0) -	(3.90e-1) -	(1.17e-1) =	**(2.76e-1)**
+/-/=			14/20/11	15/21/9	4/38/3	18/22/5	7/28/10	

图 3.3 对 WFG3 测试集上 4 个目标情况下上述算法所得的 Pareto 解集进行介绍。由图可知，GrEA、RVEA 算法得到的 Pareto 前沿分布相对较差。NSGA-III 和 MOEA/D 算法得到的分布最好，说明了在解决此类问题的分布性能。CMAPIO_KIGD 算法的分布性与 MAPIO 算法的性能相近且介于上述两类算法性能之间，也说明了所提方法在算法收敛性方面具有较强的优势。



(a) NSGA-III (b) MOEA/D

(c) GrEA (d) RVEA

(e) MAPIO (f)CMAPIO_KIGD

图 3.3 不同算法当目标个数为 4 时在 WFG3 测试函数上的 Pareto 前沿

Fig 3.3 Pareto Front obtained by different algorithms on the 4-objective WFG test function

为验证算法所提不同策略对性能的影响，在 DTLZ3、5 和 6 测试示例上进行了相应的消融实验，如表 3.4 所示。其中 CMAPIO 算法仅使用基于竞争思想的种群更新策略，MAPIO-KIGD 算法仅使用基于 KIGD 指标的环境选择策略。由对比实验结果可以看出，

单独使用一种策略的算法性能相近且要差于所提的 CMAPIO_KIGD 算法，同时实验结果也验证了所提的两种策略对算法性能的提升同等重要。

表 3.4 DTLZ 测试问题中不同目标下不同策略消融算法的 IGD 值

Table 3.4 The IGD value of different strategies for different objectives in the DTLZ test problems

Problem	M	D	MAPIO-KIGD	CMAPIO	CMAPIO_KIGD
DTLZ3	10	19	4.1816e+1 (8.92e+0) -	3.2414e+2 (3.24e+1) -	**2.0587e+0 (1.95e+0)**
	15	24	1.8909e+1 (4.44e+0) -	2.0108e+2 (1.06e+2) -	**1.4139e+0 (3.26e-1)**
DTLZ5	8	17	2.1223e-1 (1.11e-1) =	2.4839e-1 (3.12e-2) -	**8.4433e-2 (2.28e-2)**
	10	19	3.9314e-1 (7.68e-2) -	2.9520e-1 (5.33e-2) -	**9.6701e-2 (3.47e-2)**
	15	24	7.8517e-1 (8.75e-2) -	**2.8530e-1 (4.37e-2) =**	4.2488e-1 (1.79e-1)
DTLZ6	8	17	**6.7340e-1 (3.81e-1) =**	5.0413e+0 (5.36e-1) -	1.2713e+0 (7.59e-1)
	10	19	**5.3444e-1 (2.03e-1) +**	5.5970e+0 (8.87e-1) -	2.3568e+0 (6.74e-1)
	15	24	1.7763e+0 (4.42e-1) -	3.9147e+0 (5.70e-1) -	**1.1027e+0 (3.27e-1)**
+/-/=			1/5/2	0/7/1	

3.4 小结

本章首先对现有的高维多目标鸽群算法在求解高维多目标优化问题时收敛性和多样性冲突的问题进行了相应的分析。然后针对目标个数急剧增加导致算法收敛性和多样性不足的问题，考虑对感兴趣区域个体的进一步选择与分析，设计了基于 KIGD 指标的竞争高维多目标鸽群算法（CMAPIO_KIGD），利用基于 KIGD 指标的环境选择策略在 Pareto 支配对个体排序的基础上进一步进行 KIGD 排序从而保证一定收敛性的同时提高拐点区域算法的多样性。最后利用基于竞争机制的速度位置更新策略对种群中的个体进行两种不同的更新策略，失败者利用胜利者的信息进行更新，从而提高算法的收敛性性能。在标准测试集 DTLZ 和 WFG 上进行的仿真实验也进一步验证了所提 CMAPIO_KIGD 算法在收敛性和多样性方面的性能。

第四章　基于多选择策略的高维多目标鸽群算法

前两章从提高个体性能的两个方面对现有的高维多目标算法进行了相应的设计，一定程度上提高了算法在求解高维多目标问题的个体选择能力以及算法收敛性和多样性。然而，算法在解决具有不同特征的问题时性能并不确定。因此，本章旨在前两章的基础上，从并发集成的角度[85]，通过构建选择策略池，将精英个体保留在外部归档集中，从而提高高维多目标鸽群算法在解决不同特性的高维多目标优化问题的性能并通过仿真实验比较进一步分析了本章提出的算法的性能。

4.1 问题分析

表 4.1 不同特性测试问题分析

Table 4.1 Analysis of different types of test problems

测试函数	特性	分析
DTLZ1，WFG3、DTL2，DTL3，DTLZ4，WFG2、WFG4-WFG9	线性、凹面性、凸面性、混合性、连通性	Pareto 最优前沿面的形状不同，对应于目标函数为线性函数、凸函数、凹函数或者混合多种性质，以及判断 Pareto 前沿是否互相连通。
WFG2，WFG3，WFG6	可分离性、不可分离性	可分离目标可以通过逐个独立地考虑每个参数来优化，全局最优参数向量的结果集是每个单独优化参数的最优集的叉乘。相反，不可分离目标中所有的参数之间是相互关联的，不可通过参数分解单独对单个参数进行优化。
DTLZ1，DTLZ3，DTLZ4	多模态、欺骗性	存在多个 Pareto 解集映射到同一个 PF，具有多个全局或者局部 Pareto 最优解集。求解比较困难，容易陷入局部最优。欺诈性问题是特殊多模态问题，目标函数至少有两个最优，一个真实最优和一个欺骗最优。但大多数搜索空间必须偏向于欺骗最优，全局最优放到一个不太可能的地方。
WFG3，DTLZ5，DTLZ6，DTLZ7	退化性	退化前沿是一个比它所嵌入的目标空间的维数更低的前沿。例如，三个目标问题中的线段。退化Pareto最优前沿会给某些算法带来问题。例如，如果帕累托最优前沿是退化的，提高算法在前沿面均匀分布的相对性能就会较差。
DTLZ4，WFG1，WFG5，WFG7，WFG8，WFG9	偏差性	搜索空间中均匀分布的参数向量样本映射到适应度空间中存在偏差，即从Pareto最优集到Pareto最优前沿的映射是有偏差的，需要决策者判断在目标空间还是搜索空间的均匀分布。

通过前两章的分析，可以看出不同测试集上的不同测试问题具有其特定的特性针对DTLZ 和 WFG 标准测试集，进一步分析问题的特性，可以有的放矢的对算法进行改进，同时进一步的提高算法在收敛性和多样性方面的综合性能。表 4. 1 对问题特性进行了相应的分析，针对不同特性的模型以及问题，可以采取不同的策略来提高算法相应的求解性能。例如，针对偏差性问题，可以考虑基于偏好的策略提高算法的搜索性能；针对退化性问题，由于其真实 Pareto 前沿维度低于目标空间的维度，算法在均匀性方面需求相对较弱，需采用提高算法收敛性的策略。因此，基于现有高维多目标鸽群算法求解不同类型问题性能不足的原因，以及更好的提高算法在收敛性和多样性等综合性能，本章提出了基于多选择策略的高维多目标鸽群算法，进一步提高算法整体的收敛性和多样性等综合性能。

4.2 基于多选择策略的高维多目标鸽群算法

本节详细介绍了所提的基于多选择策略的高维多目标鸽群算法（MSMAPIO），通过构建基于多选择策略的选择策略池，将精英个体保留在外部归档集中从而进一步影响算法进化过程，提高算法的综合性能。首先对算法的主要框架进行介绍。然后在 4.2.2 节中对所提的基于多选择策略的精英保留机制进行详细阐述。再然后在 4.2.3 和 4.2.4 节对种群更新策略以及算法更新涉及到的外部归档集策略和其他进化策略进行简单介绍。最后在 4.2.5 节中对算法所需的时间复杂度进行了相应的分析。

4.2.1 MSMAPIO 算法主要框架

Algorithm 4.1 MSMAPIO 的总体框架

初始化种群 P，构建外部归档集 A，均匀分布参考点集 R；
初始化种群中个体速度 V_i，位置 X_i，个体 p_i 的适应度值，局部最优位置 $pbest_i$，局部中心点 P_{center}；
利用现有种群更新外部归档集 A，并计算归档集中个体适应度值；
while $T \le T_{\max}$
根据 4.2.3 节的速度位置更新对种群进行更新；
计算更新后种群个体的速度度 V_i，位置 X_i；
计算种群中个体 p_i 的适应度值，更新局部最优位置 $pbest_i$，局部中心点 P_{center}；
构建由多个选择策略组成的选择策略池；
从选择策略池中选择精英个体放到 A 中；
对外部归档集 A 进行更新，并对其执行进化策略得到新种群 S；
计算新种群 S 中个体的适应度值；
更新外部归档集 A；
end while
输出：种群 P，外部归档集 A

MSMAPIO 的总体框架如算法 4.1 所示。首先执行初始化操作。种群 P 、外部归档集 A 和均匀分布的参考点集 R 被初始化，同时初始化种群中每个鸽子个体的速度 V_i 与位置 X_i 。然后，计算现有种群的局部中心点 P_{center} 以及每个鸽子个体 p_i 的目标函数值。接着对种群中的个体进行非支配排序并将优势个体放到外部归档集 A 中，对现有归档集进行更新。然后执行主要的进化过程。通过在 4.2.3 节中所提的速度位置更新策略对种群中的个体进行更新操作得到新的种群 P 并计算其适应度值，以及种群的局部最优个体 $pbest_i$ 和局部中心个体 P_{center} 。根据 4.2.2 节的基于多选择策略的精英保留机制对外部归档集进行更新。然后，通过在外部归档集上执行模拟二进制交叉(SBX)和基于多项式的变异(PM)两种进化策略，得到一个新的种群 s 。直到循环迭代结束，得到最后的种群 P 和外部归档集 A 。

4.2.2 基于多选择策略的精英保留策略

大多数进化算法主要是对所选择的精英个体采用相应的进化策略来更新种群。当问题的复杂性增加时，这些进化策略也会导致进化过程中一些精英个体的丢失，从而导致解集的性能表现不好。因此，受多重选择策略优点的启发[86]，本文提出了一种精英个体保留策略，能够使解集在收敛性、多样性、均匀性和求解精度方面都有相对更好的表现。

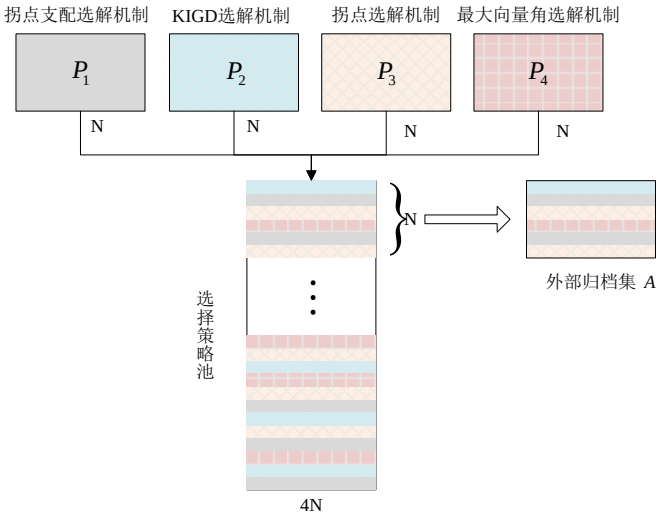


图 4.1 精英保留机制主要过程

Fig 4.1 The main procedure of elite individual retention

如图 4.1 所示，采用四种不同的选择策略从初始群体 P 中选择个体生成子代 P_1, P_2, P_3, P_4 。其中每个子代都有 N 个个体，进一步构建总数为 $4N$ 的选择策略池。然后从中选择 N 个个体更新到外部归档集 A 中从而实现精英个体的保留。在 MSMAPIO 的选择策略池中，基于拐点支配的由本文第二章提出，在策略池中用来提高算法的选择压力以及其逼近真实前沿的能力。基于 KIGD 的选择策略是第三章所提出的，用来提高局

部区域内的多样性。基于拐点的选择策略[40]可以近似地看作是对更大超体积的偏好，用来加速解的收敛。基于向量角的选择策略[56]采用最大向量角优先原则，通过找到向量角最大的解和极值解，保证了解集的多样性和均匀性。

精英个体保持策略的伪代码如算法 4.1 所示。P_1, P_2, P_3, P_4 是由四种不同选择策略所产生的子代。首先，由不同的选择算子得到相应子代，并组合成一个新的解集 S。然后，比较 A 和 S 中解之间的 Pareto 支配关系。如果 A_j 支配 S_i，则返回 1；当 S_i 支配 A_j，则返回 -1 值。最后将外部归档集 A 根据 BFE 策略[82]计算的适应度值进行更新。

算法 4.1 精英个体保留策略

开始

输入：种群 P，种群大小 N，参考点集 R，拐点集合 K，外部归档集 A；

P_1 = 拐点支配选择(P);

P_2 = KIGD 指标选择(P);

P_3 = 拐点选择(P);

P_4 = 向量角选择(P);

$S = P_1 \cup P_2 \cup P_3 \cup P_4$;

for $i = 1$ to $|S|$

for $j = 1$ to $|A|$

$teb = CheckDominance(S_i, A_j)$; /*比较个体的 Pareto 支配关系*/

if $teb == 1$

A_j 被标记为支配解；

else if $teb == -1$

S_i 被标记为支配解；

end if

end for

从归档集 A 中删除所有标记的解决方案；

if S_i is not tagged

将个体 S_i 添加到归档集 A 中；

if $|A| > N$

计算 BFE 适应度值；

根据适合度值剔除性能较差的个体；

end if

end if

end for

输出: 归档集 A

结束

（1）基于拐点支配的选择策略

拐点支配选择策略是本文第二章所提的个体选择策略，在 Pareto 支配对个体划分层级之后利用个体的拐点支配关系进一步对个体进行选择，从而提高算法对个体的选择能力。具体细节详见 2.2.3 节。

（2）基于 KIGD 指标的选解策略

基于 KIGD 指标的选解策略是本文第三章所提的选择策略，利用个体计算的 KIGD 指标值在个体 Pareto 支配排序之后对非支配个体进行进一步排序，从而在保证一定收敛性的同时提高算法的多样性。具体细节详见 3.2.2 节。

（3）基于拐点机制的选解策略

基于拐点机制的选择策略[40]采用了三种锦标赛选择策略，如支配关系、拐点优先准则和加权距离度量。拐点作为 Pareto 最优前沿的真子集，具有较大超体积的偏置，因此将这些点作为衡量种群收敛的主要标准。同时，在不知道拐点数目的情况下，对小范围内的拐点进行自适应识别。利用自适应策略定位局部区域拐点，从而加快收敛速度。

首先，利用支配关系来选择解决方案。当解不占支配地位时，进一步考虑解是否为拐点。如果两者都不是，则判断解之间的加权距离。如果加权距离相同，则随机选择一个解决方案。拐点的确定如图 4.2 所示，其中 B、G、E 分别为膝关节点。但是如果邻域的个数为 1，那么只有 E 是拐点。基于拐点的选择策略的细节参考[40]。

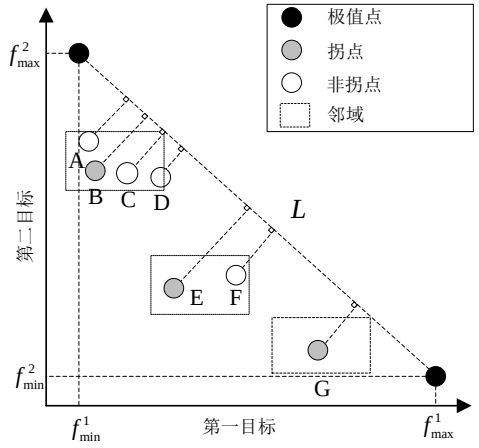


图 4.2 二维目标空间最小化问题拐点确定

Fig 4.2 Determining knee points for two objectives minimization problem

其中，加权距离度量的计算如下所示。其中种群个数为 p，邻域个数为 k。

$$DW(p)=\sum_{i=1}^{k} w_{p_i} dis_{pp_i} \tag{4.1}$$

$$w_{p_i}=\frac{r_{p_i}}{\sum_{i=1}^{k} r_{p_i}} \tag{4.2}$$

$$r_{p_i}=\frac{1}{\left|dis_{pp_i}-\frac{1}{k}\sum_{i=1}^{k} dis_{pp_i}\right|} \tag{4.3}$$

p_i是个体 p 的第 i 个最近邻个体。w_{p_i} 是个体 p_i 计算的权重。dis_{pp_i} 是个体 p 和它最近邻个体 p_i 之间的欧式距离。r_{p_i} 表示距离 dis_{pp_i} 的排序。

（4）基于向量角的选解策略

在基于向量角的选择策略[56]中，采用最大向量角优先原则来保证解集的分布。在非支配排序之后，计算每个解在归一化目标空间中的范数。然后在归一化的向量空间中计算两个解之间的向量角，从而比较个体之间的向量角，从而选出精英个体。通过寻找具有较大矢量角的解，保证了解集的多样性和均匀性。

目标空间中的解决方案 x_j 的范数定义如公式 4.4 所示。两个解之间的向量角计算如公式 4.5 所示。

$$norm\left(x_j\right)\triangleq\sqrt{\sum_{i=1}^{m} f_i^{'}\left(x_j\right)^2} \tag{4.4}$$

$$angle\left(x_j,y_k\right)\triangleq\arccos\left|\frac{F^{'}\left(x_j\right)\bullet F^{'}\left(y_k\right)}{norm\left(x_j\right)\cdot norm\left(y_k\right)}\right| \tag{4.5}$$

其中 $f_i^{'}\left(x_j\right)$ 代表 x_j 的第 i 个目标向量归一化值。$F^{'}\left(x_j\right)\bullet F^{'}\left(y_k\right)$ 是归一化目标向量 $F^{'}\left(x_j\right)$ 和 $F^{'}\left(y_k\right)$ 之间的内积。

$$F^{'}\left(x_j\right)\bullet F^{'}\left(y_k\right)=\sum_{i=1}^{m} f_i^{'}\left(x_j\right)\cdot f_i^{'}\left(y_k\right) \tag{4.6}$$

4.2.3 高维多目标鸽群算法种群更新策略

MSMAPIO 利用了 Cui 在[35]中提出的速度和位置更新方程，可以为鸽子个体提供新的搜索方向，即从中心位置点到全局最优位置方向的进化方向。提出的更新方程如下：

$$\begin{aligned}V_i\left(t_{now}\right)=e^{-Rt}\cdot V_i\left(t_{now}-1\right)+r_1\cdot r_2\cdot tr\cdot\left(1-\log_T^{t_{now}}\right)\cdot\left(X_{glo}-X_i\left(t_{now}\right)\right)+\\ r_3\cdot r_4\cdot tr\cdot\log_T^{t_{now}}\cdot\left(X_{cen}-X_i\left(t_{now}\right)\right)+r_5\cdot r_6\cdot\left(X_{glo}-X_{cen}\right)\end{aligned} \tag{4.7}$$

$$X_i\left(t_{now}\right)=X_i\left(t_{now}-1\right)+V_i\left(t_{now}\right) \tag{4.8}$$

其中 t_{now} 表示当前迭代次数，T 代表最大迭代次数，R 为地图与指南针算子，tr 是确保从地图与指南针算子平稳过渡到地标算子之间的迁移因子。X_{glo} 为个体全局最优位置，

X_{cen}为当前迭代中某些个体的中心位置信息，计算如第三章公式(3.5)。r_1， r_3和r_5表示三个属于$[0,1]$之间的随机分布，提供不同程度的随机更新；r_2， r_4和r_6表示三个学习因子。它们定义如下：

$$r_i = \begin{cases} 0, 0 < rand() \le \dfrac{1}{M} \\ 1, \dfrac{1}{M} < rand() \le 1 \end{cases} \tag{4.10}$$

其中$rand()$位于$[0,1]$之间的随机数。M代表目标空间的维度。当r_i为 0 时，公式 2.5 中该部分将不再影响速度的更新。当r_i是 1 时，随着目标维度M的更新，$V_i(t_{now})$也将动态更新。

4.2.4 外部归档集以及进化策略

在 MSMAPIO 中，采用了一个外部归档集策略[84]来保留精英解决方案，它可以为真正的 Pareto 前沿提供适当的选择。同时，利用 BFE 方法[87]，将精英个体保留在外部归档集A中，用于下一代迭代优化。新的种群可以根据与外部归档集的比较进行种群的更新。受[35]的启发，MSMAPIO 额外使用一种进化搜索模式来防止算法陷入局部最优，其中包括模拟二进制交叉(SBX) [35]和多项式变异(PM)[35]。

4.2.5 时间复杂度分析

本节主要分析所提的 MSMAPIO 算法的时间复杂度。该算法的计算复杂度主要来源于多选择策略的精英保留策略、速度位置更新策略和归档集更新策略。精英保留策略中选择策略池构建中，由前两章内容可知，基于拐点支配的选解策略、基于 KIGD 指标的选解策略的时间负责度均为$O(N^2M)$，基于拐点机制和基于向量角的选择策略对应的时间复杂度也为$O(N^2M)$。由于每个选择策略之间是独立执行的因此选择策略池构建的整体时间复杂度为$O(4N^2M)$。计算对选择策略池中个体选择的时间复杂度时，种群大小为$4N$，目标空间维度为M，因此时间复杂度为$O(4N^2M)$。速度位置更新策略对应的时间复杂度为$O(M)$。归档集更新的计算复杂度主要是通过将种群中的个体与归档集中的精英个体进行比较得到的。因为归档集的大小通常是根据种群的大小来设定的，因此最差情况下它的时间复杂度为$O(N^2M)$。综上所述，当处于最差情况下，MSMAPIO 算法的整体时间复杂度为$O(N^2M)$。

4.3 仿真实验

本节对所提 MSMAPIO 算法的性能进行进一步分析。首先，先与前两章所提方法在 DTLZ 和 WFG 测试集上进行了相应的比较。然后，与五种经典算法 NSGA-III[8]、GrEA[9]、

MOEA/D[54]、RVEA[55]和 MAPIO[35]在 DTLZ[38]和 WFG[39]测试集上进行了相应的仿真实验比较，分析了所提方法在不同测试集上的性能，进一步验证了所提 MSMAPIO 算法在求解不同类型问题时的性能。

4.3.1 参数设置及评价指标

本章实验的测试集以及对比算法的设置同前两章相同，因此参照前两章参数进行相应设置。

4.3.2 与前两章算法的实验结果比较与分析

本节首先将所提算法跟前两章所提的 KnMAPIO 与 CMAPIO_IGD 算法在 WFG 测试集上进行了相应的比较，并利用 IGD 指标对算法性能进行了相应的衡量。如表 4.2 所示，加粗黑体显示的值表示算法求得的最优解，“+”、“-”和“=”表示其他算法性能优于、劣于或者等于所提的 MSMAPIO 算法。从表格的最后一行可以看出，跟 KnMAPIO 算法相比，本章所提的 MSMAPIO 算法在 10 个测试问题上性能相对较差，但在 15 个测试问题上占据显著优势，在 20 个测试问题中所提算法与 KnMAPIO 算法性能相近，没有显著性差异。而与 CMAPIO_KIGD 算法的比较中，所提的算法性能更加显著，在 17 个测试问题上所提算法的性能更好，而仅在 5 个问题中算法性能要劣于 CMAPIO_KIGD。对于 WFG1、WFG2、WFG3 测试问题上，所提方法的性能显著较好，证明其在求解可分离单模态问题上有着相对较好的性能。

表 4.2 在 WFG 测试集不同目标下与前两章算法比较的 IGD 值

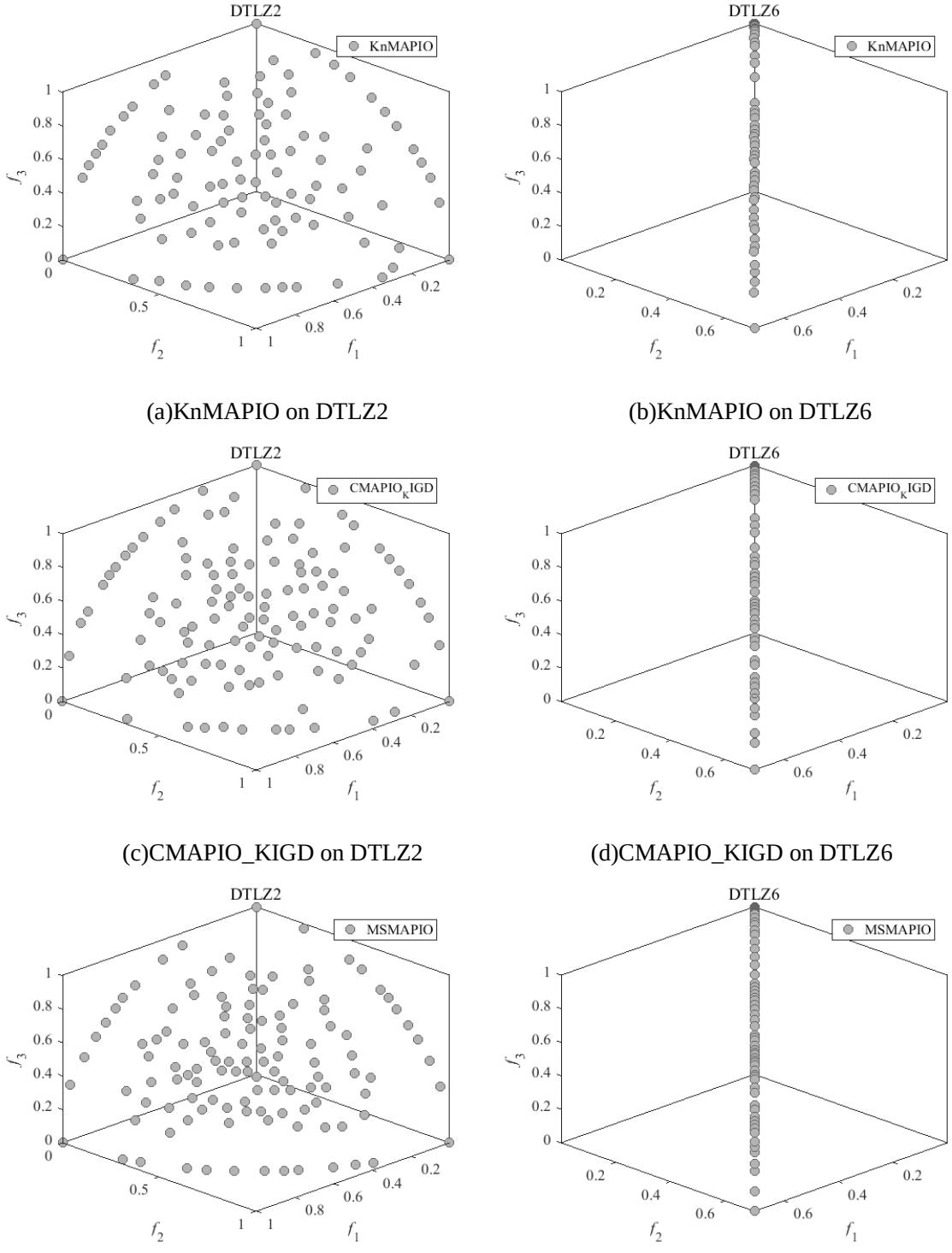
Table 4.3 IGD values compared with the algorithms in the previous two chapters under different objectives in the WFG test function

Problem	M	D	KnMAPIO	CMAPIO_KIGD	MSMAPIO
WFG1	4	13	1.8231e+0 (2.61e-2) -	1.8857e+0 (7.94e-2) -	**1.8174e+0 (9.19e-2)**
	6	15	2.3096e+0 (8.64e-2) =	2.3340e+0 (7.16e-2) =	**2.2603e+0 (9.62e-2)**
	8	17	2.7409e+0 (9.15e-2) -	2.7569e+0 (4.85e-2) =	**2.7153e+0 (4.69e-2)**
	10	19	3.1379e+0 (3.02e-2) =	**3.1108e+0 (3.85e-2) =**	3.1246e+0 (6.28e-2)
	15	24	**3.9560e+0 (1.28e-1) =**	4.0373e+0 (9.53e-2) =	4.0333e+0 (7.83e-2)
WFG2	4	13	5.3559e-1 (5.42e-2) -	5.7791e-1 (6.87e-2) -	**4.7624e-1 (3.22e-2)**
	6	15	8.3714e-1 (8.94e-2) -	8.3512e-1 (6.67e-2) -	**7.4392e-1 (5.33e-2)**
	8	17	1.3091e+0 (9.08e-2) -	1.3340e+0 (9.88e-2) -	**1.1134e+0 (4.84e-2)**
	10	19	**1.2451e+0 (6.19e-2) =**	1.4569e+0 (8.54e-2) -	1.2876e+0 (7.20e-2)
	15	24	**2.3738e+0 (2.09e-1) +**	2.7200e+0 (2.82e-1) =	2.6684e+0 (1.77e-1)
WFG3	4	13	**2.4470e-1 (3.36e-2) =**	5.8162e-1 (3.37e-1) -	2.6792e-1 (2.50e-2)
	6	15	5.6360e-1 (5.18e-2) -	8.0822e-1 (5.18e-1) =	**5.5622e-1 (4.26e-2)**
	8	17	8.9614e-1 (7.50e-2) =	1.0051e+0 (3.50e-1) =	**8.5325e-1 (1.30e-1)**
	10	19	1.1065e+0 (1.36e-1) =	1.7403e+0 (6.81e-1) =	**9.8637e-1 (1.22e-1)**

	15	24	2.1813e+0 (1.77e+0) -	2.4976e+0 (1.12e+0) -	**9.9972e-1 (1.34e-1)**
WFG4	4	13	7.0124e-1 (1.70e-2) =	**6.7711e-1 (1.94e-2) =**	6.9702e-1 (1.88e-2)
	6	15	1.8020e+0 (1.46e-2) -	**1.7589e+0 (1.84e-2) =**	1.7707e+0 (1.87e-2)
	8	17	3.0287e+0 (2.71e-2) =	3.0309e+0 (4.06e-2) =	**3.0286e+0 (3.76e-2)**
	10	19	4.2479e+0 (2.15e-2) -	4.2532e+0 (5.17e-2) -	**4.1602e+0 (4.04e-2)**
	15	24	**8.4525e+0 (1.54e-1) +**	8.6205e+0 (9.16e-2) =	8.6573e+0 (7.13e-2)
WFG5	4	13	6.7817e-1 (1.38e-2) =	**6.6636e-1 (2.23e-2) =**	6.6845e-1 (2.58e-2)
	6	15	1.7508e+0 (1.82e-2) =	1.7439e+0 (2.48e-2) =	**1.7357e+0 (2.33e-2)**
	8	17	**2.9921e+0 (2.72e-2) +**	3.0738e+0 (3.60e-2) -	3.0181e+0 (3.25e-2)
	10	19	**4.0826e+0 (2.37e-2) +**	4.3794e+0 (6.65e-2) -	4.2112e+0 (5.98e-2)
	15	24	**8.5133e+0 (6.97e-2) +**	8.5197e+0 (1.29e-1) +	8.7479e+0 (9.02e-2)
WFG6	4	13	7.8118e-1 (1.91e-2) -	**6.9921e-1 (1.81e-2) =**	7.2330e-1 (2.68e-2)
	6	15	1.9224e+0 (2.91e-2) -	**1.8381e+0 (3.15e-2) =**	1.8405e+0 (4.56e-2)
	8	17	3.2014e+0 (3.82e-2) =	3.1886e+0 (5.49e-2) =	**3.1667e+0 (2.66e-2)**
	10	19	4.2032e+0 (1.35e-2) +	4.3429e+0 (8.12e-2) -	4.2636e+0 (5.16e-2)
	15	24	8.7889e+0 (9.22e-2) -	8.6206e+0 (1.78e-1) =	**8.6187e+0 (1.02e-1)**
WFG7	4	13	7.1925e-1 (1.14e-2) -	**6.7747e-1 (1.40e-2) =**	6.8679e-1 (1.35e-2)
	6	15	**1.7811e+0 (2.43e-2) =**	1.7899e+0 (2.78e-2) =	1.7945e+0 (3.03e-2)
	8	17	3.0581e+0 (3.81e-2) =	3.1063e+0 (5.18e-2) -	**3.0371e+0 (4.00e-2)**
	10	19	4.2842e+0 (2.71e-2) -	4.6107e+0 (2.14e-1) -	**4.2133e+0 (4.11e-2)**
	15	24	**8.6314e+0 (2.27e-1) +**	9.2198e+0 (2.82e-1) =	9.2548e+0 (4.52e-1)
WFG8	4	13	**7.2805e-1 (1.50e-2) =**	9.3366e-1 (5.96e-2) -	7.3999e-1 (1.16e-2)
	6	15	**1.8703e+0 (1.44e-2) =**	2.1234e+0 (5.47e-2) -	1.8851e+0 (2.80e-2)
	8	17	3.3825e+0 (2.06e-2) =	3.5334e+0 (6.19e-2) -	**3.3807e+0 (2.84e-2)**
	10	19	4.6489e+0 (5.06e-2) -	4.9249e+0 (9.19e-2) -	**4.5434e+0 (4.61e-2)**
	15	24	8.9238e+0 (2.98e-1) +	**8.6370e+0 (3.74e-1) +**	9.2776e+0 (3.83e-1)
WFG9	4	13	6.3942e-1 (1.51e-2) =	6.3837e-1 (9.18e-3) =	**6.3630e-1 (1.32e-2)**
	6	15	1.7083e+0 (2.45e-2) =	**1.6935e+0 (2.19e-2) =**	1.6962e+0 (1.13e-2)
	8	17	3.0817e+0 (4.63e-2) =	**3.0308e+0 (3.06e-2) +**	3.0853e+0 (2.43e-2)
	10	19	**4.1874e+0 (7.23e-2) +**	4.2442e+0 (8.22e-2) +	4.3492e+0 (4.94e-2)
	15	24	**8.8531e+0 (1.50e-1) +**	8.9243e+0 (1.07e-1) +	9.1019e+0 (1.36e-1)
+/-/=			10/15/20	5/17/23	

图 4. 3 对所提算法跟前两章所提的 KnMAPIO 与 CMAPIO_IGD 算法在目标维度为 3 维的 DTLZ2 和 DTLZ6 测试问题上产生的 Pareto 前沿进行了相应的展示。如图 4.3 所示，图（a）、图（c）、图（e）为 KnMAPIO、CMAPIO_KIGD 与本文所提算法在 DTLZ2 测试问题中的 Pareto 前沿图，可以看出，在 DTLZ2 测试问题中算法的分布性具有相似的性质。但是相比较而言，本文所提的分布性与其他两个算法相比性能相对较好。图（b）、图（d）、图（f）为 KnMAPIO、CMAPIO_KIGD 与本文所提算法在 DTLZ6 测试问题中的 Pareto 前沿图，从图中可以看出，KnMAPIO 算法对应的算法分布性最差，CMAPIO_KIGD 性能居中，而所提的 MSMAPIO 算法的分布性最好。而对于 MSMAPIO

较好的分布性能，所提的基于多选择策略的精英个体保留策略以及选择策略中采用的参考点策略均起到了一定的作用。



(a)KnMAPIO on DTLZ2　　(b)KnMAPIO on DTLZ6

(c)CMAPIO_KIGD on DTLZ2　　(d)CMAPIO_KIGD on DTLZ6

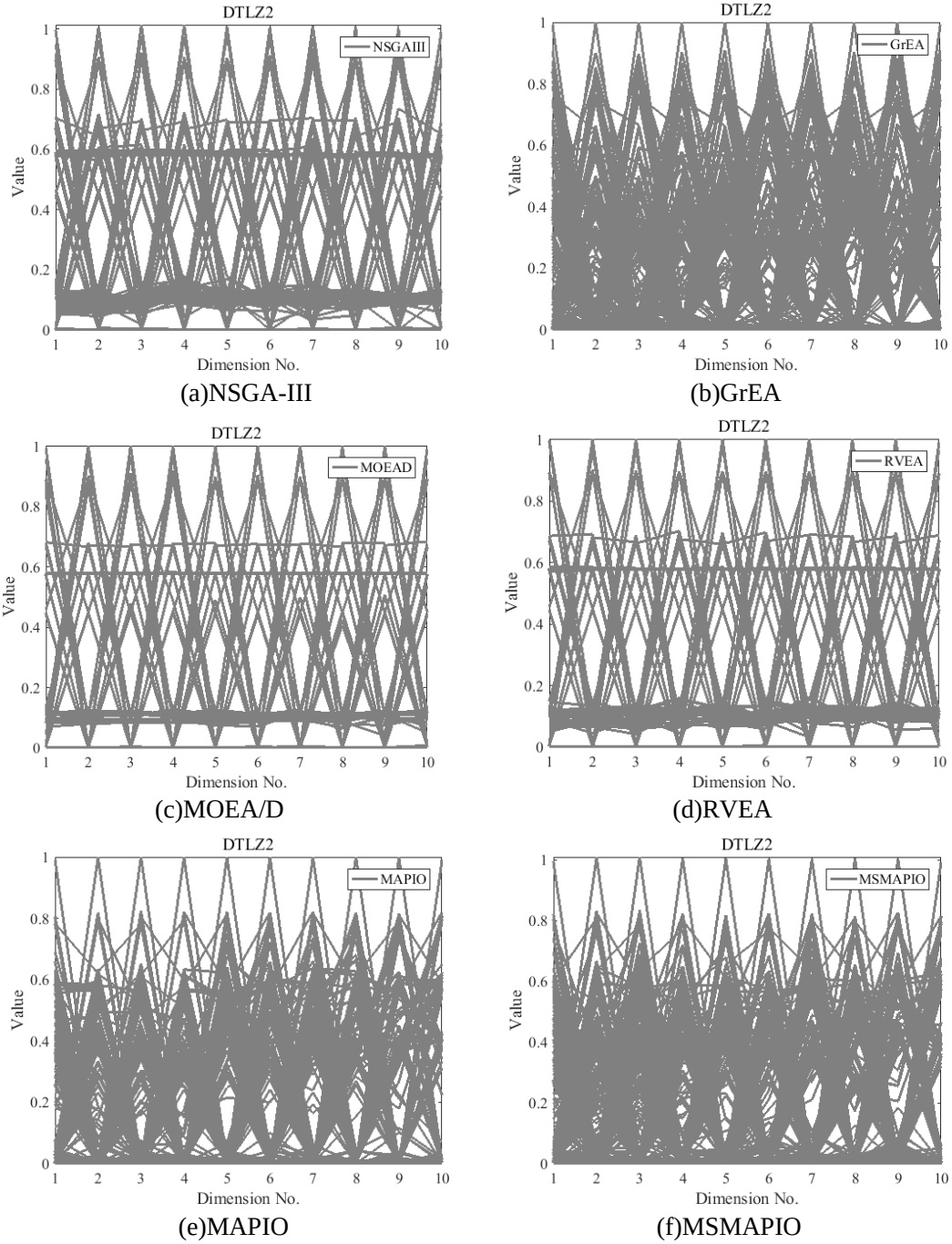
(e)MSMAPIO on DTLZ2　　(f)MSMAPIO on DTLZ6

图 4.3 三种算法当目标个数为 3 时在 DTLZ2 和 DTLZ6 测试函数上的 Pareto 前沿

Fig 4.3 Pareto Front obtained by three algorithms on the 3-objective DTLZ2 and DTLZ6 test function

4.3.3 在 DTLZ 标准测试集上的实验结果及分析

为了测试 MSMAPIO 算法的性能，将所提出的算法与其他五种多目标优化算法在 DTLZ 测试函数上进行了比较，并通过 IGD 指标对算法结果进行了评价。在 DTLZ2 测试函数上，上述算法得到的 Pareto 解集如图 4.4 所示。由图可知，MOEA/D、NSGA-III 和 RVEA 得到的 Pareto 前沿分布相对较差。GrEA 和 MAPIO 算法得到的分布比较良好，说明这两种算法的综合性能。MSMAPIO 算法的分布性能介于两者之间。



(a)NSGA-III (b)GrEA

(c)MOEA/D (d)RVEA

(e)MAPIO (f)MSMAPIO

图 4.4 不同算法当目标个数为 10 时在 DTLZ2 测试函数上的 Pareto 前沿

Fig 4.4 Pareto Front obtained by different algorithms on the 10-objective DTLZ test function

表 4.3 显示了所提的 MSMAPIO 与 GrEA、NSGA-III、RVEA、MOEA/D 和 MAPIO 五种最先进算法在 DTLZ 测试函数上 IGD 指标的比较结果。从表 4.3 最后一行可以看出，在 40 个测试函数中，MSMAPIO 与 NSGA-III 和 GrEA 相比具有明显的优势，NSGA-III 在任何函数上的表现都不比其他函数好，而 GrEA 与本文算法相比只在一个测试问题上有最好的结果。与 MOEA/D 和 RVEA 的比较结果中，MOEA/D 在 14 个测试问题中优于本文算法，而 RVEA 在 8 个测试问题中优于 MSMAPIO 算法。MOEA/D 算法在 DTLZ1、DTLZ3 和 DTLZ7 测试函数上的性能优于其他算法，这是因为测试函数具有多模态的特点，且难以获得收敛解，基于分解的 MOEA/D 算法在求解这类问题上具有较好的性能。在与 MAPIO 的比较结果中，本文算法有轻微的优势，有 9 个测试问题上比 MAPIO 差，但在 11 个测试问题中比它要好。突出显示区域显示了 MSMAPIO 算法在 DTLZ2, DTLZ4 和 DTLZ8 上的优越性能。DTLZ2 的 Pareto 前沿面是凹面的，也证明了所提方法求解凹面性问题的性能。

表 4.3 六种算法在 DTLZ 测试问题上不同目标下的 IGD 值

Table 4.3 The IGD values of six algorithms for different objectives in the DTLZ test problems

Problem	M	NSGAIII	GrEA	MOEAD	RVEA	MAPIO	ImMAPIO
DTLZ1	4	2.3888e+1 (5.83e+0) -	2.4071e+1 (7.48e+0) -	**7.2175e+0 (3.10e+0) +**	1.9722e+1 (5.57e+0) =	1.6976e+1 (5.99e+0) =	1.9585e+1 (5.62e+0)
	6	2.0401e+1 (7.24e+0) =	2.4501e+1 (7.27e+0) =	**5.8150e+0 (2.53e+0) +**	1.6796e+1 (4.97e+0) +	1.8277e+1 (4.61e+0) +	2.2822e+1 (7.40e+0)
	8	2.5337e+1 (7.80e+0) =	2.3780e+1 (7.10e+0) =	**6.2681e+0 (2.98e+0) +**	1.7858e+1 (4.56e+0) +	1.7862e+1 (5.85e+0) +	2.3125e+1 (6.09e+0)
	10	2.7201e+1 (8.48e+0) -	2.6984e+1 (6.26e+0) -	**7.3240e+0 (3.16e+0) +**	2.3775e+1 (7.71e+0) =	1.9401e+1 (5.50e+0) =	2.0198e+1 (5.22e+0)
	15	2.2261e+1 (7.88e+0) =	2.3769e+1 (7.69e+0) =	**7.3268e+0 (3.23e+0) +**	1.2768e+1 (3.39e+0) +	1.6969e+1 (4.65e+0) +	2.2674e+1 (6.23e+0)
DTLZ2	4	5.0397e-1 (3.18e-2) -	4.9442e-1 (3.47e-2) -	4.3144e-1 (4.43e-2) -	5.2941e-1 (4.44e-2) -	3.5691e-1 (2.30e-2) -	**3.2650e-1 (2.14e-2)**
	6	7.3475e-1 (3.40e-2) -	7.1729e-1 (3.87e-2) -	7.1453e-1 (8.24e-2) -	7.5866e-1 (4.37e-2) -	5.5658e-1 (2.88e-2) -	**5.0884e-1 (2.66e-2)**
	8	9.0014e-1 (4.31e-2) -	8.6907e-1 (4.56e-2) -	8.2620e-1 (9.30e-2) -	9.3744e-1 (3.85e-2) -	7.3065e-1 (3.08e-2) -	**6.7541e-1 (3.50e-2)**
	10	9.9208e-1 (4.50e-2) =	9.9122e-1 (3.73e-2) =	1.0046e+0 (3.35e-2) =	9.9227e-1 (2.87e-2) =	**9.8746e-1 (3.82e-2) =**	9.9872e-1 (2.80e-2)
	15	1.2018e+0 (3.46e-2) -	1.1669e+0 (3.58e-2) -	1.1952e+0 (7.96e-2) -	1.2790e+0 (5.74e-2) -	1.0439e+0 (2.58e-2) -	**1.0212e+0 (2.77e-2)**
DTLZ3	4	2.6493e+2 (5.65e+1) =	2.4989e+2 (3.76e+1) +	**7.5101e+1 (1.75e+1) +**	2.3674e+2 (3.36e+1) +	1.8886e+2 (3.75e+1) +	2.8646e+2 (5.91e+1)

	6	2.9861e+2 (4.81e+1) =	2.9476e+2 (5.29e+1) =	**6.1431e+1 (1.38e+1) +**	2.3572e+2 (5.58e+1) +	2.3187e+2 (4.00e+1) +	2.7578e+2 (5.85e+1)
	8	3.0952e+2 (6.44e+1) -	3.6041e+2 (5.95e+1) -	**7.0422e+1 (2.00e+1) +**	2.2531e+2 (4.34e+1) +	2.4397e+2 (4.01e+1) +	2.6897e+2 (3.95e+1)
	10	3.7231e+2 (5.99e+1) -	3.6206e+2 (5.61e+1) -	**7.0760e+1 (2.39e+1) +**	3.2356e+2 (5.69e+1) =	2.4865e+2 (3.56e+1) +	2.9963e+2 (5.72e+1)
	15	3.5092e+2 (7.94e+1) -	3.6350e+2 (7.67e+1) -	**7.9687e+1 (1.82e+1) +**	2.0457e+2 (5.01e+1) +	2.2626e+2 (3.74e+1) +	2.9846e+2 (5.63e+1)
DTLZ4	4	8.7855e-1 (8.38e-2) -	8.0189e-1 (9.52e-2) -	8.8553e-1 (2.03e-1) -	7.4879e-1 (7.54e-2) =	7.5976e-1 (9.77e-2) =	**7.4544e-1 (1.04e-1)**
	6	1.0763e+0 (7.93e-2) -	1.0064e+0 (8.25e-2) -	1.1349e+0 (1.00e-1) -	1.0685e+0 (8.44e-2) -	8.8936e-1 (6.75e-2) =	**8.8272e-1 (8.88e-2)**
	8	1.1102e+0 (7.31e-2) -	1.0409e+0 (5.48e-2) -	1.1979e+0 (1.07e-1) -	1.1272e+0 (7.53e-2) -	9.4858e-1 (6.68e-2) =	**9.1424e-1 (6.48e-2)**
	10	1.1442e+0 (6.09e-2) =	**1.1291e+0 (6.08e-2) =**	1.1370e+0 (5.25e-2) =	1.1403e+0 (5.81e-2) =	1.1425e+0 (3.92e-2) =	1.1444e+0 (5.00e-2)
	15	1.2471e+0 (5.36e-2) -	1.1985e+0 (5.23e-2) -	1.3014e+0 (7.10e-2) -	1.2929e+0 (5.56e-2) -	**1.0667e+0 (4.07e-2) =**	1.0749e+0 (5.46e-2)
DTLZ5	4	3.4967e-1 (5.29e-2) -	3.4604e-1 (3.51e-2) -	2.3627e-1 (6.94e-2) -	4.2477e-1 (5.39e-2) -	**1.4400e-1 (5.83e-2) +**	1.7401e-1 (2.20e-2)
	6	3.8873e-1 (3.61e-2) -	3.8807e-1 (4.20e-2) -	3.0491e-1 (6.37e-2) -	5.0149e-1 (5.89e-2) -	1.8638e-1 (3.78e-2) =	**1.8392e-1 (2.50e-2)**
	8	3.6485e-1 (3.96e-2) -	3.8458e-1 (4.21e-2) -	2.7851e-1 (8.50e-2) -	5.4130e-1 (6.42e-2) -	**1.8872e-1 (2.78e-2) =**	1.9267e-1 (2.47e-2)
	10	3.6992e-1 (3.74e-2) =	3.6597e-1 (3.67e-2) =	3.8113e-1 (3.00e-2) =	**3.5890e-1 (3.77e-2) =**	3.7866e-1 (3.49e-2) =	3.7017e-1 (3.42e-2)
	15	3.9242e-1 (4.33e-2) -	3.8907e-1 (5.12e-2) -	3.2872e-1 (8.10e-2) -	5.6115e-1 (7.91e-2) -	2.0885e-1 (2.98e-2) =	**2.0679e-1 (3.60e-2)**
DTLZ6	4	8.0477e+0 (2.14e-1) -	7.9284e+0 (2.17e-1) -	7.8025e+0 (4.67e-1) -	8.0799e+0 (3.18e-1) -	**6.4606e+0 (7.11e-1) =**	6.6087e+0 (5.38e-1)
	6	8.2582e+0 (1.91e-1) -	8.1236e+0 (1.45e-1) -	7.7522e+0 (5.25e-1) -	8.3874e+0 (2.73e-1) -	6.6396e+0 (6.90e-1) =	**6.6123e+0 (5.19e-1)**
	8	8.1601e+0 (1.80e-1) -	8.1142e+0 (1.87e-1) -	7.4586e+0 (6.07e-1) -	8.4815e+0 (2.30e-1) -	**6.3743e+0 (6.18e-1) =**	6.3994e+0 (6.41e-1)
	10	8.3975e+0 (1.54e-1) =	8.3974e+0 (9.50e-2) =	8.3814e+0 (1.32e-1) =	**8.3201e+0 (1.68e-1) =**	8.3756e+0 (1.09e-1) =	8.3867e+0 (1.13e-1)
	15	8.2145e+0 (2.42e-1) -	8.1845e+0 (2.33e-1) -	7.1694e+0 (5.63e-1) -	8.5950e+0 (4.14e-1) -	6.7692e+0 (6.39e-1) -	**6.3664e+0 (7.91e-1)**
DTLZ7	4	1.0721e+1 (9.39e-1) =	1.0523e+1 (8.97e-1) =	**6.8686e+0 (1.51e+0) +**	9.0019e+0 (8.53e-1) +	1.1170e+1 (1.12e+0) -	1.0333e+1 (7.84e-1)
	6	1.7522e+1 (1.41e+0) -	1.7168e+1 (1.25e+0) -	**1.2474e+1 (1.92e+0) +**	1.6993e+1 (1.70e+0) -	1.6776e+1 (1.79e+0) -	1.5615e+1 (1.70e+0)
	8	2.3403e+1	2.3105e+1	**1.6554e+1**	2.2673e+1	2.2713e+1	2.1635e+1

	10	(1.63e+0) - 2.8954e+1	(2.46e+0) - **2.8632e+1**	**(3.01e+0) +** 2.9084e+1	(1.94e+0) - 2.9295e+1	(1.78e+0) = 2.9114e+1	(2.22e+0) 2.8991e+1
	15	(1.83e+0) = 4.8157e+1	**(2.23e+0) =** 4.7771e+1	(2.09e+0) = **3.7765e+1**	(1.82e+0) = 4.6075e+1	(2.52e+0) = 4.8246e+1	(2.84e+0) 4.6251e+1
		(3.39e+0) -	(3.24e+0) -	**(5.13e+0) +**	(3.23e+0) =	(3.13e+0) -	(2.95e+0)
DTLZ8	4	2.6816e-1 (1.20e-2) -	2.5290e-1 (1.66e-2) -	2.9764e-1 (3.31e-2) -	2.5702e-1 (1.79e-2) -	2.5491e-1 (2.05e-2) -	**2.3448e-1 (1.91e-2)**
	6	3.2956e-1 (1.36e-2) -	3.2875e-1 (1.60e-2) -	3.7389e-1 (3.52e-2) -	3.5488e-1 (2.08e-2) -	3.1313e-1 (1.76e-2) -	**2.9977e-1 (1.69e-2)**
	8	3.8214e-1 (1.36e-2) -	3.7347e-1 (1.79e-2) -	4.3741e-1 (4.44e-2) -	4.1455e-1 (1.95e-2) -	3.5998e-1 (1.64e-2) -	**3.4724e-1 (1.28e-2)**
	10	**4.0439e-1 (1.03e-2) =**	4.0760e-1 (1.28e-2) =	4.0825e-1 (1.33e-2) =	4.1142e-1 (1.06e-2) =	4.1032e-1 (1.25e-2) =	4.0898e-1 (1.14e-2)
	15	5.0885e-1 (1.47e-2) -	5.1728e-1 (1.20e-2) -	6.0916e-1 (4.75e-2) -	5.6736e-1 (2.08e-2) -	4.9992e-1 (1.52e-2) =	**4.9466e-1 (1.88e-2)**
+/-/=		0/28/12	1/28/11	14/20/6	8/21/11	9/11/20	

4.3.4 在 WFG 标准测试集上的实验结果及分析

为了检验所提算法的性能，表 4.4 和表 4.5 将 MSMAPIO 算法与其他五种包括 MOEA/D、NSGA-III、RVEA、GrEA 和 MAPIO 的多目标优化算法在 WFG 测试函数上进行比较，并用 IGD 和 HV 对实验结果进行评价。从表 4.4 的最后一行可以看出，与 NSGA-III 相比，MSMAPIO 在 45 个测试函数上有着明显的优势，而 NSGA-III 与本文算法相比只有一个最好的结果。这是因为使用 Pareto 支配关系和参考点策略在目标个数急剧增加时缺乏选择压力，从而导致较差的收敛性能。在与 GrEA、MOEA/D 和 RVEA 的比较结果中，GrEA 在 8 个测试问题中优于所提的 MSMAPIO 算法，MOEA/D 在 6 个测试问题中性能优于本文算法，RVEA 则仅在 4 个问题上优于本文算法。在与 MAPIO 的比较结果中，本文提出的算法也具有一定的优势。这是因为我们提出的 MSMAPIO 算法不仅采用了和 MAPIO 算法相同的 BFE 策略，同时采用的基于多选择策略的精英个体保留机制可以将更多性能较好的个体保留在归档集中，这也为提高种群的收敛性和多样性提供一定的帮助。虽然本文算法在 WFG1 和 WFG2 测试问题上的性能没有超过其他算法，但在整个测试函数集上，尤其是在 WFG6、WFG7 和 WFG9 测试函数上的性能优于其他算法。MSMAPIO 中基于拐点支配的的选择策略能够在提高算法选择压力的同时提高其逼近真实 Pareto 前沿的能力。基于 KIGD 指标的选解策略能够保证算法在局部区域中的多样性。基于拐点机制的选择策略可以提高种群的收敛性。而基于矢量角度的选择策略可以保证算法的广泛性和均匀性。综上，基于多选择策略的精英个体保留机制为个体的选择提供了更多的可能，从而提高了算法求解多种类型问题的综合性能。

表 4.4 六种算法在 WFG 测试问题上不同目标的 IGD 值

Table 4.4 The IGD values of six algorithms for different objectives in the WFG test problems

Problem	M	NSGAIII	GrEA	MOEAD	RVEA	MAPIO	ImMAPIO
WFG1	4	2.2679e+0 (5.83e-2) +	2.0836e+0 (6.07e-2) +	**1.9482e+0 (9.49e-2) +**	2.1434e+0 (1.60e-1) +	2.1668e+0 (5.32e-2) +	2.3239e+0 (4.25e-2)
	6	2.6632e+0 (3.62e-2) =	2.5207e+0 (4.50e-2) +	**2.3985e+0 (3.50e-2) +**	2.7377e+0 (1.51e-1) =	2.5629e+0 (2.97e-2) +	2.6873e+0 (4.78e-2)
	8	3.0568e+0 (3.73e-2) =	2.9148e+0 (4.07e-2) +	**2.8456e+0 (3.58e-2) +**	3.0935e+0 (1.22e-1) =	2.9695e+0 (5.04e-2) +	3.0501e+0 (5.50e-2)
	10	3.3744e+0 (2.58e-2) =	3.3030e+0 (3.10e-2) +	**3.2350e+0 (3.25e-2) +**	3.4821e+0 (8.31e-2) -	3.3225e+0 (2.93e-2) +	3.3813e+0 (3.10e-2)
	15	4.3397e+0 (6.01e-2) =	4.2716e+0 (5.94e-2) +	**4.2636e+0 (3.39e-2) +**	4.3775e+0 (6.75e-2) -	4.2776e+0 (5.32e-2) +	4.3155e+0 (3.48e-2)
WFG2	4	8.8248e-1 (6.15e-2) -	8.1748e-1 (1.00e-1) -	9.7502e-1 (1.55e-1) -	8.4418e-1 (1.13e-1) -	**5.7956e-1 (5.57e-2) +**	7.3098e-1 (7.18e-2)
	6	1.4167e+0 (2.03e-1) =	1.1910e+0 (1.14e-1) +	1.5752e+0 (2.99e-1) =	1.2032e+0 (1.56e-1) +	**1.1057e+0 (2.22e-1) +**	1.4936e+0 (2.46e-1)
	8	2.1908e+0 (3.82e-1) =	1.7794e+0 (2.27e-1) +	2.4856e+0 (5.20e-1) -	**1.7235e+0 (1.70e-1) +**	1.7806e+0 (2.64e-1) +	2.1205e+0 (3.58e-1)
	10	3.3382e+0 (4.79e-1) -	2.9570e+0 (5.79e-1) =	2.6705e+0 (6.97e-1) =	2.7563e+0 (4.26e-1) =	**2.5374e+0 (3.46e-1) =**	2.6705e+0 (3.92e-1)
	15	4.7815e+0 (1.14e+0) =	3.8352e+0 (8.18e-1) +	4.4897e+0 (1.49e+0) +	**3.8040e+0 (6.68e-1) +**	4.4664e+0 (7.12e-1) +	5.0460e+0 (8.25e-1)
WFG3	4	7.6595e-1 (3.04e-2) -	7.5258e-1 (5.13e-2) -	1.2244e+0 (2.43e-1) -	8.7538e-1 (4.80e-2) -	6.1213e-1 (3.97e-2) -	**5.7753e-1 (3.57e-2)**
	6	1.1042e+0 (4.48e-2) -	1.0629e+0 (5.37e-2) -	1.8104e+0 (3.36e-1) -	1.5754e+0 (9.76e-2) -	8.1342e-1 (3.78e-2) -	**7.7909e-1 (3.96e-2)**
	8	1.3299e+0 (6.67e-2) -	1.3116e+0 (6.40e-2) -	2.4667e+0 (3.43e-1) -	2.2462e+0 (1.56e-1) -	9.5307e-1 (5.42e-2) =	**9.3275e-1 (5.69e-2)**
	10	1.5305e+0 (6.61e-2) =	**1.5074e+0 (6.31e-2) =**	1.5220e+0 (6.56e-2) =	1.5149e+0 (6.00e-2) =	1.5104e+0 (5.34e-2) =	1.5172e+0 (6.61e-2)
	15	2.3159e+0 (1.12e-1) -	2.1354e+0 (8.42e-2) -	5.3943e+0 (1.40e+0) -	4.9843e+0 (2.88e-1) -	1.5755e+0 (9.29e-2) =	**1.5687e+0 (1.44e-1)**
WFG4	4	1.2495e+0 (9.74e-2) -	1.0623e+0 (9.81e-2) -	1.3239e+0 (1.53e-1) -	1.1744e+0 (1.18e-1) -	1.0430e+0 (1.03e-1) -	**9.5073e-1 (7.55e-2)**
	6	3.5989e+0 (2.80e-1) -	3.4923e+0 (2.35e-1) -	3.6606e+0 (4.57e-1) -	3.9492e+0 (3.52e-1) -	2.9753e+0 (2.94e-1) -	**2.7154e+0 (2.41e-1)**
	8	6.5429e+0 (3.66e-1) -	6.3809e+0 (3.94e-1) -	6.0238e+0 (4.31e-1) -	7.1237e+0 (4.26e-1) -	5.4316e+0 (4.31e-1) =	**5.3036e+0 (4.05e-1)**
	10	**9.3553e+0 (4.51e-1) =**	9.5364e+0 (3.67e-1) =	9.4802e+0 (3.87e-1) =	9.5797e+0 (4.14e-1) =	9.4157e+0 (4.38e-1) =	9.5146e+0 (5.16e-1)

	15	1.8761e+1 (7.33e-1) -	1.7837e+1 (7.92e-1) -	1.8129e+1 (1.12e+0) -	1.9520e+1 (9.30e-1) -	1.7050e+1 (1.12e+0) =	**1.6813e+1 (9.02e-1)**
WF G5	4	1.1252e+0 (4.47e-2) -	1.0532e+0 (4.12e-2) -	1.5054e+0 (2.21e-1) -	1.1365e+0 (3.27e-2) -	1.0562e+0 (3.54e-2) -	**9.8621e-1 (3.16e-2)**
	6	2.7896e+0 (1.11e-1) -	2.6278e+0 (1.29e-1) -	3.2877e+0 (2.31e-1) -	2.8435e+0 (1.41e-1) -	2.3348e+0 (8.44e-2) -	**2.2062e+0 (6.07e-2)**
	8	5.0017e+0 (2.60e-1) -	4.7340e+0 (2.40e-1) -	5.5699e+0 (3.87e-1) -	5.2512e+0 (2.32e-1) -	4.1651e+0 (1.11e-1) -	**4.0178e+0 (1.10e-1)**
	10	7.2526e+0 (2.88e-1) =	**7.1248e+0 (3.24e-1) =**	7.2314e+0 (3.42e-1) =	7.2017e+0 (2.96e-1) =	7.1677e+0 (3.12e-1) =	7.2006e+0 (2.65e-1)
	15	1.4940e+1 (6.03e-1) -	1.5364e+1 (6.09e-1) -	1.5001e+1 (1.06e+0) -	1.6095e+1 (8.23e-1) -	1.3166e+1 (5.92e-1) -	**1.2696e+1 (5.08e-1)**
WF G6	4	1.2505e+0 (4.23e-2) -	1.2093e+0 (4.29e-2) -	1.6584e+0 (2.31e-1) -	1.3003e+0 (6.09e-2) -	1.1608e+0 (4.65e-2) -	**1.0867e+0 (4.38e-2)**
	6	2.9531e+0 (1.75e-1) -	2.8090e+0 (1.03e-1) -	3.4805e+0 (3.13e-1) -	3.1188e+0 (1.47e-1) -	2.6019e+0 (1.29e-1) -	**2.4118e+0 (9.83e-2)**
	8	5.2762e+0 (2.94e-1) -	5.0758e+0 (2.36e-1) -	5.6672e+0 (2.72e-1) -	5.6237e+0 (3.36e-1) -	4.7682e+0 (3.18e-1) -	**4.4482e+0 (1.99e-1)**
	10	7.4959e+0 (2.30e-1) =	7.5062e+0 (2.62e-1) =	7.5112e+0 (3.37e-1) =	7.6682e+0 (3.84e-1) -	7.5592e+0 (3.05e-1) =	**7.4866e+0 (3.07e-1)**
	15	1.5541e+1 (8.42e-1) -	1.5833e+1 (6.47e-1) -	1.5868e+1 (7.99e-1) -	1.6171e+1 (7.16e-1) -	1.4975e+1 (7.09e-1) -	**1.4041e+1 (5.77e-1)**
WF G7	4	1.1391e+0 (6.44e-2) -	1.0519e+0 (6.12e-2) -	1.4513e+0 (2.11e-1) -	1.1253e+0 (5.14e-2) -	1.0057e+0 (7.45e-2) -	**9.3171e-1 (4.16e-2)**
	6	3.0335e+0 (2.01e-1) -	2.8505e+0 (1.58e-1) -	3.4183e+0 (2.34e-1) -	3.2046e+0 (2.32e-1) -	2.6170e+0 (1.81e-1) -	**2.3913e+0 (1.49e-1)**
	8	5.4840e+0 (3.35e-1) -	5.2294e+0 (3.19e-1) -	5.8570e+0 (3.46e-1) -	5.8238e+0 (4.26e-1) -	5.0230e+0 (3.23e-1) -	**4.6235e+0 (2.15e-1)**
	10	8.0380e+0 (3.84e-1) =	8.0994e+0 (3.17e-1) =	8.1133e+0 (4.20e-1) =	8.1266e+0 (3.52e-1) =	8.0862e+0 (3.99e-1) =	**7.9913e+0 (3.60e-1)**
	15	1.6222e+1 (7.83e-1) -	1.6131e+1 (6.31e-1) -	1.6466e+1 (9.98e-1) -	1.7083e+1 (6.68e-1) -	1.5289e+1 (7.77e-1) -	**1.4755e+1 (6.68e-1)**
WF G8	4	1.3602e+0 (4.34e-2) -	1.3191e+0 (4.73e-2) -	1.6783e+0 (1.20e-1) -	1.4033e+0 (6.55e-2) -	1.3062e+0 (5.37e-2) -	**1.2182e+0 (5.06e-2)**
	6	3.1347e+0 (1.80e-1) -	3.0351e+0 (1.31e-1) -	3.5691e+0 (2.42e-1) -	3.3104e+0 (1.51e-1) -	2.8526e+0 (1.67e-1) -	**2.6735e+0 (1.10e-1)**
	8	5.4695e+0 (2.57e-1) -	5.2240e+0 (1.88e-1) -	5.9009e+0 (3.95e-1) -	5.8838e+0 (3.13e-1) -	5.1078e+0 (2.78e-1) -	**4.7267e+0 (1.94e-1)**
	10	7.7556e+0 (2.50e-1) =	7.7605e+0 (3.37e-1) =	7.9148e+0 (3.35e-1) =	**7.7397e+0 (3.23e-1) =**	7.7631e+0 (3.05e-1) =	7.7501e+0 (3.06e-1)
	15	1.5751e+1 (6.14e-1) -	1.5860e+1 (5.12e-1) -	1.6241e+1 (8.39e-1) -	1.6671e+1 (8.24e-1) -	1.5230e+1 (8.01e-1) -	**1.4330e+1 (6.44e-1)**
WF	4	1.2771e+0	1.2225e+0	1.5791e+0	1.3426e+0	1.1065e+0	**1.0018e+0**

G9	6	(6.50e-2) - 3.1110e+0	(6.66e-2) - 3.0923e+0	(1.05e-1) - 3.5141e+0	(7.02e-2) - 3.3101e+0	(7.48e-2) - 2.5172e+0	**(4.89e-2) 2.2930e+0**
	8	(1.82e-1) - 5.5229e+0	(2.10e-1) - 5.3238e+0	(3.75e-1) - 5.6756e+0	(1.79e-1) - 5.9342e+0	(1.51e-1) - 4.4527e+0	**(1.06e-1) 4.2249e+0**
	10	(3.33e-1) - 8.0942e+0	(2.83e-1) - 8.0332e+0	(3.51e-1) - 8.0114e+0	(3.48e-1) - 8.0706e+0	(2.51e-1) - 7.9967e+0	**(1.89e-1) 7.9617e+0**
	15	(2.79e-1) = 1.6040e+1	(3.66e-1) = 1.5619e+1	(3.93e-1) = 1.5887e+1	(3.09e-1) = 1.6794e+1	(4.35e-1) = 1.3647e+1	**(4.08e-1) 1.2918e+1**
		(6.28e-1) -	(8.42e-1) -	(1.09e+0) -	(6.36e-1) -	(7.31e-1) -	**(5.12e-1)**
+/-/=		1/30/14	8/29/8	6/30/9	4/32/9	9/24/12	

表 4.5 显示了 MSMAPIO 算法与其他五种算法在 4、6、8、10、15 个目标的 WFG 测试函数上的 HV 值的仿真实验结果。采用相同的分析方法，将 NSGA-III、GrEA、MOEA/D、RVEA 和 MAPIO 与 MSMAPIO 算法进行比较，可以看出 MSMAPIO 算法具有明显的优势。在 WFG1 上，GrEA 算法的性能优于其他算法。这是因为 GrEA 中的网格划分和网格支配机制在解决单峰问题时可以通过对目标空间的有效划分而提高目标空间的搜索性能。而 MAPIO 在 WFG2 测试功能上的显著性能也为我们未来求解不可分问题提供新的思路。在 WFG4-WFG9 测试函数上，所提 MSMAPIO 算法的性能显著优于其他算法。从数值分析的 45 个对比结果中，所提的 MSMAPIO 在 31 个测试问题上优于 NSGA-III，优于 GrEA 的有 28 个测试问题，优于 RVEA 和 MOEA/D 的结果有 30 个，优于 MAPIO 的结果有 25 个。加粗显示的部分也证实了所提算法在 WFG4-WFG9 上的优越性能。上述两个仿真实验的对比结果均验证了所提算法在整体 WFG 测试函数上的优越性，也进一步验证了所提策略在求解不同类型问题时的有效性。

表 4.5 六种算法在 WFG 测试问题上不同目标的 HV 值

Table 4.5 The HV values of these algorithms for different objectives in the WFG test problems

Problem	M	NSGAIII	GrEA	MOEAD	RVEA	MAPIO	MSMAPIO
WFG1	4	3.5784e+1 (1.85e+1) +	9.0046e+1 (2.00e+1) +	9.2829e+1 (2.28e+1) +	**1.1726e+2 (1.70e+1) +**	6.3304e+1 (1.74e+1) +	2.0385e+1 (1.26e+1)
	6	1.1416e+4 (1.86e+3) +	**1.6261e+4 (1.76e+3) +**	1.3574e+4 (1.35e+3) +	1.4804e+4 (2.12e+3) +	1.4192e+4 (1.52e+3) +	8.2635e+3 (2.81e+3)
	8	3.6132e+6 (3.59e+5) +	**4.5193e+6 (1.58e+5) +**	3.5048e+6 (2.91e+5) +	4.0264e+6 (4.85e+5) +	4.0921e+6 (3.18e+5) +	2.9339e+6 (9.62e+5)
	10	1.4806e+9 (1.06e+8) +	1.6884e+9 (6.38e+7) +	1.3266e+9 (1.38e+8) +	1.3983e+9 (1.70e+8) +	**1.7054e+9 (5.03e+7) +**	9.0170e+8 (4.47e+8)
	15	2.4364e+16 (6.79e+14) +	**2.5234e+16 (2.39e+14) +**	1.8951e+16 (2.23e+15) +	2.3302e+16 (2.09e+15) +	2.0558e+16 (5.80e+15) +	9.9738e+15 (3.37e+15)
WFG2	4	3.8408e+2 (1.20e+1) -	3.9749e+2 (1.17e+1) -	3.4158e+2 (2.62e+1) -	3.8459e+2 (1.84e+1) -	**4.3437e+2 (1.16e+1) +**	4.1143e+2 (1.02e+1)

	6	5.3681e+4 (2.12e+3) -	5.6442e+4 (1.72e+3) =	4.8562e+4 (3.52e+3) -	5.3773e+4 (2.71e+3) -	**6.2267e+4 (2.38e+3) +**	5.5325e+4 (2.36e+3)
	8	1.3448e+7 (5.10e+5) -	1.4286e+7 (6.13e+5) =	1.2110e+7 (1.31e+6) -	1.3357e+7 (7.98e+5) -	**1.5963e+7 (4.86e+5) +**	1.4333e+7 (4.39e+5)
	10	5.2616e+9 (1.65e+8) -	5.4782e+9 (2.41e+8) -	5.0647e+9 (5.88e+8) -	5.3277e+9 (3.26e+8) -	**6.5392e+9 (1.61e+8) +**	6.1226e+9 (1.49e+8)
	15	8.3501e+16 (4.60e+15) -	8.9301e+16 (5.56e+15) =	7.6236e+16 (8.10e+15) -	8.3628e+16 (5.65e+15) -	**1.0295e+17 (3.56e+15) +**	8.9246e+16 (3.00e+15)
WFG3	4	6.7255e-1 (2.21e-1) -	6.2814e-1 (2.41e-1) -	2.6620e-2 (5.10e-2) -	4.0052e-1 (1.89e-1) -	9.8456e-1 (2.56e-1) -	**1.1281e+0 (2.64e-1)**
	6	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
	8	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
	10	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
	15	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	**0.0000e+0 (0.00e+0)**
WFG4	4	2.0090e+2 (7.17e+0) -	2.1494e+2 (8.23e+0) -	1.6090e+2 (1.96e+1) -	1.9753e+2 (1.07e+1) -	2.3180e+2 (1.10e+1) -	**2.4395e+2 (8.49e+0)**
	6	2.8430e+4 (1.41e+3) -	2.9478e+4 (1.13e+3) -	2.1901e+4 (1.72e+3) -	2.2975e+4 (1.45e+3) -	3.5850e+4 (1.77e+3) -	**3.7948e+4 (1.74e+3)**
	8	7.8951e+6 (2.75e+5) -	8.2707e+6 (3.00e+5) -	6.0156e+6 (5.60e+5) -	6.0796e+6 (2.62e+5) -	1.0200e+7 (4.00e+5) -	**1.0723e+7 (4.02e+5)**
	10	**3.4974e+9 (1.07e+8) =**	3.4541e+9 (1.07e+8) =	3.4326e+9 (1.16e+8) =	3.4557e+9 (8.69e+7) =	3.4728e+9 (9.23e+7) =	3.4472e+9 (1.26e+8)
	15	6.0675e+16 (3.05e+15) -	6.4184e+16 (2.64e+15) -	4.6306e+16 (5.62e+15) -	4.3322e+16 (3.82e+15) -	7.7883e+16 (3.94e+15) -	**8.0952e+16 (3.10e+15)**
WFG5	4	1.6439e+2 (6.53e+0) -	1.7568e+2 (6.79e+0) -	1.0502e+2 (1.86e+1) -	1.5588e+2 (7.28e+0) -	1.7808e+2 (6.93e+0) -	**1.9221e+2 (5.20e+0)**
	6	2.5761e+4 (8.79e+2) -	2.6921e+4 (8.49e+2) -	1.4994e+4 (2.97e+3) -	2.1958e+4 (7.74e+2) -	2.9153e+4 (1.05e+3) -	**3.0877e+4 (1.16e+3)**
	8	7.2302e+6 (2.71e+5) -	7.6177e+6 (1.66e+5) -	4.1603e+6 (9.29e+5) -	5.8004e+6 (2.75e+5) -	8.2373e+6 (3.02e+5) -	**8.6090e+6 (2.83e+5)**
	10	3.2628e+9 (6.43e+7) -	3.2956e+9 (7.80e+7) =	3.2963e+9 (8.58e+7) =	3.2922e+9 (6.50e+7) =	3.3015e+9 (6.51e+7) =	**3.3026e+9 (7.18e+7)**
	15	5.4309e+16 (1.58e+15) -	5.6930e+16 (2.02e+15) -	3.0704e+16 (6.49e+15) -	3.7635e+16 (2.52e+15) -	6.3712e+16 (2.51e+15) -	**6.6755e+16 (2.17e+15)**
WFG6	4	1.4221e+2 (6.04e+0) -	1.4936e+2 (6.11e+0) -	8.2764e+1 (2.10e+1) -	1.2882e+2 (8.17e+0) -	1.6224e+2 (7.44e+0) -	**1.7343e+2 (6.17e+0)**
	6	2.2630e+4 (1.01e+3) -	2.3855e+4 (6.79e+2) -	1.2621e+4 (3.04e+3) -	1.8782e+4 (1.10e+3) -	2.6545e+4 (8.94e+2) -	**2.8445e+4 (8.91e+2)**
	8	6.3329e+6	6.7261e+6	3.7067e+6	5.0147e+6	7.5557e+6	**7.9697e+6**

	10	(2.32e+5) - 2.8954e+9 (1.04e+8) =	(2.09e+5) - 2.8886e+9 (6.31e+7) =	(6.93e+5) - 2.8932e+9 (9.69e+7) =	(3.09e+5) - 2.8775e+9 (9.96e+7) =	(2.38e+5) - 2.8897e+9 (9.88e+7) =	**(2.67e+5) 2.9020e+9 (1.02e+8)**
	15	4.7881e+16 (1.77e+15) -	4.9365e+16 (1.98e+15) -	2.7847e+16 (3.99e+15) -	3.3242e+16 (2.58e+15) -	5.7045e+16 (2.36e+15) -	**6.1368e+16 (2.21e+15)**
WFG7	4	1.8153e+2 (5.66e+0) -	1.9349e+2 (7.23e+0) -	1.2537e+2 (2.25e+1) -	1.7471e+2 (6.68e+0) -	2.1006e+2 (6.05e+0) -	**2.2167e+2 (4.70e+0)**
	6	2.7572e+4 (1.28e+3) -	2.8581e+4 (9.34e+2) -	1.7851e+4 (2.80e+3) -	2.2908e+4 (9.57e+2) -	3.2718e+4 (1.20e+3) -	**3.4878e+4 (1.10e+3)**
	8	7.7367e+6 (2.74e+5) -	8.0545e+6 (2.48e+5) -	4.6446e+6 (8.49e+5) -	6.1752e+6 (3.61e+5) -	9.0819e+6 (3.77e+5) -	**9.6743e+6 (3.31e+5)**
	10	3.3529e+9 (1.26e+8) =	3.3313e+9 (9.17e+7) -	3.3678e+9 (1.07e+8) =	3.3659e+9 (9.66e+7) =	3.3725e+9 (1.02e+8) =	**3.3945e+9 (8.94e+7)**
	15	5.8204e+16 (2.31e+15) -	6.1402e+16 (2.61e+15) -	3.8224e+16 (7.41e+15) -	4.2571e+16 (2.34e+15) -	7.1340e+16 (3.73e+15) -	**7.3846e+16 (2.51e+15)**
WFG8	4	1.4275e+2 (4.75e+0) -	1.5113e+2 (5.78e+0) -	8.1614e+1 (1.51e+1) -	1.3064e+2 (7.23e+0) -	1.5802e+2 (7.83e+0) -	**1.6932e+2 (6.04e+0)**
	6	2.3171e+4 (8.35e+2) -	2.4173e+4 (9.54e+2) -	1.2178e+4 (3.21e+3) -	1.9003e+4 (1.33e+3) -	2.6918e+4 (1.05e+3) -	**2.8651e+4 (9.73e+2)**
	8	6.7731e+6 (2.14e+5) -	7.0675e+6 (2.12e+5) -	3.4656e+6 (7.54e+5) -	5.1390e+6 (3.03e+5) -	7.8044e+6 (3.40e+5) -	**8.3515e+6 (2.46e+5)**
	10	**3.1409e+9 (8.31e+7) =**	3.1283e+9 (8.30e+7) =	3.1125e+9 (9.62e+7) =	3.1360e+9 (9.03e+7) =	3.1194e+9 (7.38e+7) =	3.1146e+9 (9.06e+7)
	15	5.2570e+16 (2.44e+15) -	5.5063e+16 (2.76e+15) -	2.8691e+16 (5.63e+15) -	3.5071e+16 (3.14e+15) -	6.3728e+16 (2.62e+15) -	**6.7598e+16 (2.00e+15)**
WFG9	4	1.5209e+2 (8.27e+0) -	1.5941e+2 (1.05e+1) -	8.9853e+1 (1.76e+1) -	1.3817e+2 (1.13e+1) -	1.8435e+2 (1.12e+1) -	**2.0232e+2 (8.42e+0)**
	6	2.2504e+4 (1.27e+3) -	2.2896e+4 (1.51e+3) -	1.2910e+4 (2.91e+3) -	1.8568e+4 (1.65e+3) -	2.8914e+4 (1.92e+3) -	**3.1961e+4 (1.79e+3)**
	8	6.4729e+6 (3.35e+5) -	6.7877e+6 (3.83e+5) -	3.4978e+6 (9.81e+5) -	4.8800e+6 (4.10e+5) -	8.2628e+6 (4.38e+5) -	**8.9906e+6 (3.56e+5)**
	10	2.8154e+9 (1.42e+8) =	2.7936e+9 (1.28e+8) =	2.7844e+9 (1.06e+8) =	2.8074e+9 (1.24e+8) =	2.8082e+9 (1.15e+8) =	**2.8361e+9 (1.13e+8)**
	15	4.8525e+16 (2.74e+15) -	5.1927e+16 (3.02e+15) -	2.9578e+16 (5.41e+15) -	3.1632e+16 (2.68e+15) -	6.2604e+16 (3.23e+15) -	**6.9644e+16 (2.61e+15)**
+/-/=		5/31/9	5/28/12	5/30/10	5/30/10	10/25/10	

4.3.5 算法仿真结果的 Wilcoxon 符号秩检验

Wilcoxon 符号秩检验[88]，又称符号秩和检验，基本思想时假设配对比较的两个结果处理性能相同，即其差值的总体分布是对称分布，且其差值的总体中位数为 0。如果显著性差值小于 0.05，则拒绝原假设，认为匹配样本之间存在显著性差异。本文采用 Wilcoxon 符号秩检验来检验所提出的 MSMAPIO 算法与其他五种算法在 WFG 测试函数

上 IGD 指标的差异。检验结果如表 4.6 所示，所提算法跟其他算法的对比样本显著性差异的值显著小于 0.5，有的值甚至为 0。在前两节的仿真对比结果可以明显看出本文所提算法的测试结果要优于其他算法，在此情况下通过检验结果证明比较的数据之间存在显著差异，从而进一步证明了本文所提算法明显优于其他算法。

表 4.6 MSMAPIO 与 5 种算法的比较结果检验结果

Table 4.6 The inspection results about the comparison results between the MSMAPIO and five algorithms

Test statistics [a]					
	MSMAPIO - NSGAIII	MSMAPIO - GrEA	MSMAPIO - MOEAD	MSMAPIO - RVEA	MSMAPIO - MAPIO
Progressive significance (Double tail)	0.000	0.000	0.000	0.000	0.005
a. Wilcoxon signed-rank test					

4.4 本章小结

本章提出了一个基于多选择策略的高维多目标鸽群算法（MSMAPIO）。通过设计的基于多选择策略的精英个体保留机制，将由不同选择策略产生的精英个体通过 BFE 策略保留在外部归档集中，从而进一步利用精英个体的信息进一步影响种群的进化方向。同时也可以针对不同问题的特性选择不同策略产生的个体，从而提高在求解该类问题时算法收敛性和多样性等方面的性能。在与前两章设计算法的仿真实验性能比较分析验证了所提方法性能。同时，通过对所提 MSMAPIO 算法与 5 种经典多目标优化进化算法在 DTLZ 和 WFG 测试集上的实验结果以及对比实验的 Wilcoxon 符号秩检验结果的分析，进一步验证所提方法在求解不同特性问题上的卓越性能。

第五章 总结与展望

5.1 工作总结

鸽群算法作为最近几年提出的新型群智能优化算法，基于其原理简单，算法鲁棒性较强的特性，被众多专家学者广泛研究。而随着现有工业问题变得越来越复杂，求解高维多目标优化问题变得越发重要，日渐成为各位专家学者研究的重点。现有的高维多目标鸽群算法在一定程度上弥补了研究上的空缺，但仍存在选择压力不足，收敛性和多样性平衡不好以及解决多种类型问题性能不充分等方面问题。因此，针对上述问题，本文提出了如下改进的高维多目标鸽群算法：

1）基于拐点支配的高维多目标鸽群算法 KnMAPIO。现有高维多目标鸽群算法采用 Pareto 支配机制进行个体的选择，但当目标维度增加时，传统的 Pareto 机制对个体的选择已经不再充分，算法迭代后期，基本上所有的个体之间都是非支配的，也会进一步影响算法的收敛性能。针对现有高维多目标鸽群算法个体选择压力不足以及无法逼近真实 Pareto 前沿的问题，本文通过基于拐点支配的环境选择策略在 Pareto 支配对种群中的个体划分层级之后引入拐点支配机制从而提高算法的选择能力，使其从对整个目标空间的搜索变成对局部感兴趣拐点区域个体的选择，从而提高算法的选择压力。同时引入结合不同分布的速度位置更新策略，在种群迭代的不同阶段引入不同的分布从而提高目标空间搜索的搜索能力。两类在 PMOPs 测试集与传统 DTLZ 和 WFG 测试集上的仿真实验，也进一步验证所提算法在提高个体选择压力以及逼近真实 Pareto 前沿的能力。

2）基于 KIGD 指标的竞争高维多目标鸽群算法 CMAPIO_KIGD。当目标空间维度增加时，不仅算法选择压力会降低，种群的收敛性和多样性之间也越发冲突，而如何更好的平衡算法在求解高维多目标问题的收敛性和多样性成为亟待解决的问题。针对现有高维多目标鸽群算法在求解高维多目标优化问题时收敛性和多样性难以平衡的问题，本文设计了基于 KIGD 的环境选择策略在 Paerto 支配的基础上引入了 KIGD 排序选解策略，在保证算法一定收敛性的同时提高算法在局部区域内分布的多样性。同时引入了基于竞争机制的种群更新策略来提高算法的收敛性能，竞争成功者采用基于局部中心个体等信息进行进化，而失败者则利用成功者的已有进行个体更新。在 DTLZ 和 WFG 测试集上进行了相应的仿真对比实验，而对实验结果的分析也进一步验证了所提算法在平衡收敛性和多样性以及提高综合性能的能力。

3）基于多选择策略的高维多目标鸽群算法 MSMAPIO。上述两章内容针对现有高维多目标鸽群算法存在的问题从两个方面进行了相应的改进，现实问题具有复杂多变的特征，对应于 DTLZ 和 WFG 测试集中的不同测试问题，上述算法的适用性存在一定问

题。而针对求解不同特性测试函数性能不足的问题，本文提出了基于多选择策略的精英保留策略，通过构建基于不同选择策略的选择策略池，将精英个体通过 BFE 策略保留在外部归档集中，在解决不同类型的问题时，针对性的选择相应算子对应的个体，从而提高算法在求解该类型问题上的性能。通过对 DTLZ 和 WFG 测试集上算法性能的分析，在与其它五种经典算法的比较也证明了所提算法的性能。

5.2 展望

本文首先对现有高维多目标鸽群算法进行了相应的性能分析，针对现有高维多目标算法存在的问题从两个不同的方面进行改进，设计了相应的改进算法，最后又结合前两章的内容针对不同特性的问题设计了基于多选择策略的高维多目标算法，均是在理论方面对现有鸽群算法进行了相应的研究，改进策略也多是从算法性能本身进行考虑，在以后的研究中可以考虑结合数学分析以及相应的理论验证，从而提高相关研究的专业性。

在未来的研究工作中，也将针对实际问题以及实际工业模型进行相应的设计，并利用上述设计的算法进行求解，进一步将鸽群算法的应用扩展到各个领域中。

参 考 文 献

[1] 洪晓翠, 段礼祥, 杨晓光, 黄谦. 智能优化算法在机械故障诊断领域的应用综述[J]. 测控技术, 2021, 40(7):1-8.

[2] C. Wang, and G. H. Zhao. Decomposition and adaptive weight adjustment method with biogeography/complex algorithm for many-objective optimization[J]. PLoS ONE, 2020, 15(10):e0240131.

[3] N. Srinivas, and K. Deb. Multi objective optimization using nondominated sorting in genetic algorithms[J]. Evolutionary Computation, 1995, 2(3):221-248.

[4] A. E Smith. Multi objective optimization using evolutionary algorithms[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(5):526 526.

[5] 陈磊. 多目标进化算法理论、算法设计与应用研究[D]. 广州:广东工业大学, 2019.

[6] Z. H. Cui, and J. J. Zhang. A hybrid many-objective optimization algorithm for coal green production problem[J]. Concurrency and Computation: Practice and Experience, 2020, 33(6):e6040.

[7] 董明刚, 刘宝, 敬超. 不规则优化问题中基于动态资源分配的高维多目标优化算法（英文）[J]. Frontiers of Information Technology & Electronic Engineering, 2020, 21(8):1171-1191.

[8] K. Deb, and H. Jain. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(4):577-601.

[9] S. X. Yang, M. Q. Li, X. H. Liu, and J. H. Zheng. A Grid-Based Evolutionary Algorithm for Many-Objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2013, 17(5):721-736.

[10] H. B. Duan, and P. X. Qiao. Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning[J]. International Journal of Intelligent Computing and Cybernetics, 2014, 7(1):24-37.

[11] W. Zheng, H. M. Sun, and H. B. Duan. Protein Secondary Structure Prediction via Pigeon-Inspired Optimization*[C]. Proceedings of 2016 IEEE Chinese Guidance, Navigation and Control Conference, Nanjing, China, 2016:1934-1938.

[12] 白晨, 姚李孝, 曹雯. 基于鸽群优化算法的含分布式电源配电网状态估计[J]. 西安理工大学学报, 2018, 34(3):294-298.

[13] 胡春鹤, 王依帆, 朱书豪, 刘文定. 基于鸽群优化算法的图像分割方法研究[J]. 郑州大学学报(工学版), 2019, 40(4):42-47.

[14] 唐悦, 杨犇, 刘燕斌, 吉晓亮, 陈柏屹. 空天飞行器性能评估中的等效拟配方法研究[J]. 战术导弹技术, 2021(4):43-51.

[15] C. Li, and H. B. Duan. Target detection approach for UAVs via improved Pigeon-inspired Optimization and Edge Potential Function[J]. Aerospace Science and Technology, 2014, 39:352-360.

[16] D. F. Zhang, H. B. Duan, and Y. J. Yang. Active disturbance rejection control for small unmanned helicopters via levy flight-based pigeon-inspired optimization[J]. Aircraft Engineering and Aerospace Technology, 2016, 89(6):946-952.

[17] 杨之元, 段海滨, 范彦铭. 基于莱维飞行鸽群优化的仿雁群无人机编队控制器设计[J]. 中国科学: 技术科学, 2018, 48:161-169.

[18] 段海滨, 邱华鑫, 范彦铭. 基于捕食逃逸鸽群优化的无人机紧密编队协同控制[J]. 中国科学: 技术科学, 2015, 45:559－572.

[19] B. Zhang, and H. B. Duan. Three-Dimensional Path Planning for Uninhabited Combat Aerial Vehicle Based on Predator-Prey Pigeon-Inspired Optimization in Dynamic Environment[J]. IEEE/ACM Transactions on Computational Biology & Bioinformatics, 2017, 14(1):97-107.

[20] Z. Y. Yang，H. B. Duan，Y. M. Fan, and Y. M. Deng. Automatic Carrier Landing System multilayer parameter design based on Cauchy Mutation Pigeon-Inspired Optimization[J]. Aerospace Science and Technology, 2018, 79:518-530.

[21] 段海滨, 杨之元. 基于柯西变异鸽群优化的大型民用飞机滚动时域控制[J]. 中国科学: 技术科学[J], 2018, 48:277-288.

[22] 胡明月, 张慧芬, 苗淑平, 杨帆, 曲振明. 基于改进鸽群算法的台区三相负荷不平衡相序调整方法[J/OL]. 中国电力: 1-7 [2022-04-16].

[23] X. X. Sun, J. S. Pan, S. C. Chu, P. Hu, and A. Q. Tian. A novel pigeon-inspired optimization with QUasi-Affine TRansformation evolutionary algorithm for DV-Hop in wireless sensor networks[J]. International Journal of Distributed Sensor Networks, 2020, 16(6):155014772093274.

[24] A. Q. Tian，S. C. Chu，J. S. Pan，H. Cui, and W. M. Zhen. A Compact Pigeon-Inspired Optimization for Maximum Short-Term Generation Mode in Cascade Hydroelectric Power Station[J]. Sustainability, 2020, 12(3):1-19.

[25] X. S. Hai, Z. L. Wang, Q. Feng, Y. Ren, B. Sun, and D. Z. Yang. A novel adaptive pigeon-inspired optimization algorithm based on evolutionary game theory[L], Sci China Tech Sci, 2021, 64(3):139203.

[26] L. Yan, B. Y. Qu, Y. S. Zhu, B. H. Qiao, and P. Nagaratnam. Dynamic economic emission dispatch based on multi-objective pigeon-inspired optimization with double disturbance[J]. Science China (Information Sciences), 2019, 62(7):108-119.

[27] W. Y. Ruan, and H. B. Duan. Multi-UAV obstacle avoidance control via multi-objective social learning pigeon-inspired optimization[J]. Frontiers of Information Technology & Electronic Engineering, 2020, 21(5):740-748.

[28] G. G. Chen, J. Qian, Z. Z. Zhang, and S. Y. Li. Application of modified pigeon-inspired optimization algorithm and constraint-objective sorting rule on multi-objective optimal power flow problem[J]. Applied soft Computing, 2020, 92:106321.

[29] H. X. Qiu, and H. B. Duan. Multi-objective pigeon-inspired optimization for brushless direct current motor parameter design[J]. Sci China Tech Sci, 2015, 58:1915-1923.

[30] X. Y. Fu, F. T. S. Chan, B. Niu, N. S. H. Chung, and T. Qu. A multi-objective pigeon inspired optimization algorithm for fuzzy production scheduling problem considering mould maintenance[J]. 中国科学：信息科学（英文版）, 2019, 62(7):11-28.

[31] H. B. Duan, M. Z. Huo, and Y. H. Shi. Limit-Cycle-Based Mutant Multi-objective Pigeon-Inspired Optimization[J]. IEEE Transactions on Evolutionary Computation, 2020, 24(5):948-959.

[32] H. X. Qiu, and H. B. Duan. A multi-objective pigeon-inspired optimization approach to UAV distributed flocking among obstacles[J], Information Sciences, 2020, 509:515-529.

[33] J. L. Shang, Y. T. Li, Y. Sun, F. Li, Y. Y. Zhang, and J. X. Liu. MOPIO: A Multi-Objective Pigeon-Inspired Optimization Algorithm for Community Detection[J]. Symmetry, 2021, 13(1):49.

[34] B. D. Tong, L. Chen, and H. B. Duan. A path planning method for UAVs based on multi-objective pigeon-inspired optimisation and differential evolution[J]. International Journal of Bio-Inspired Computation, 2021, 17(2):105.

[35] Z. H. Cui, J. J. Zhang, Y. C. Wang, Y. Cao, and X. J. Cai. A pigeon-inspired optimization algorithm for many-objective optimization problems[J]. Science China (Information Sciences),2019,62(7):131-138.

[36] K. Deb, and S. Gupta. Understanding knee points in bicriteria problems and their implications as preferred solution principles[J]. Engineering Optimization, 2011, 43(11):1175-1204.

[37] G. Yu, Y. Jin, and M. Olhofer. Benchmark Problems and Performance Indicators for Search of Knee Points in Multiobjective Optimization[J]. IEEE Transactions on Cybernetics, 2020, 50(8):3531–3544.

[38] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems Evolutionary Computation[C]. Proceeding of Congress on IEEE (CEC'02), Honolulu, USA, 2002: 825-830.

[39] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(5):477-506.

[40] X. Zhang, Y. Tian, and Y. Jin. A Knee Point-Driven Evolutionary Algorithm for Many-Objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2015, 19(6):761-776.

[41] G. Yu, Y. Jin, and M. Olhofer. An a priori knee identification multi-objective evolutionary algorithm based on α-dominance[C]. Proceeding of the Genetic and Evolutionary Computation Conference Companion, New York, NY, USA, 2019: 241–242.

[42] M. Köppen, R. Vicentegarcia, and B. Nickolay. Fuzzy-Pareto-Dominance and its Application in Evolutionary Multi-objective Optimization[C]. Proceeding of International Conference on Evolutionary Multi-Criterion Optimization: Evolutionary Multi-Criterion Optimization (EMO 2005), Berlin, Heidelberg, 2005: 399-412.

[43] Y. Yuan, H. Xu, B. Wang, and X. Yao. A New Dominance Relation-Based Evolutionary Algorithm for Many-Objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2016, 20(1):16-37.

[44] L. B. Said, S. Bechikh, and K. Ghedira. The r-Dominance: A New Dominance Relation for Interactive Evolutionary Multicriteria Decision Making[J]. IEEE Transactions on Evolutionary Computation, 2010, 14(5):801-818.

[45] M. Li, S. Yang, and X. Liu. Shift-Based Density Estimation for Pareto-Based Algorithms in Many-Objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(3):348-365.

[46] L. B. Said, S. Bechikh, and K. Gh ́edira. The r-dominance: a new dominance relation for interactive evolutionary multicriteria decision making[J]. IEEE Transactions on Evolutionary Computation, 2010, 14(5):801–818.

[47] J. Molina, L. V. Santana, A. G. Hern ́andez-D´ıaz, C. A. C. Coello, and R. Caballero. g-dominance: Reference point based dominance for multiobjective metaheuristics[J]. European Journal of Operational Research, 2009, 197(2):685-692.

[48] L. Thiele, K. Miettinen, P. J. Korhonen, and J. Molina. A preference-based evolutionary algorithm for multi-objective optimization[J]. Evolutionary Computation, 2009, 17(3):411–436.

[49] G. Yu, Y. Jin, and M. Olhofer. A Multiobjective Evolutionary Algorithm for Finding Knee Regions Using Two Localized Dominance Relationships[J]. IEEE Transactions on Evolutionary Computation 2021, 25(1):145-158.

[50] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2):182-197.

[51] N. Kumar, M. S. Rahman, A. Duary, S. K. Mahato, and A. K. Bhunia. A new QPSO based hybrid algorithm for bound-constrained optimisation problem and its application in engineering design problems[J]. International Journal of Computing Science and Mathematics, 2020, 12(4):385-412.

[52] H. B. Duan, and Z. Y. Yang. Large civil aircraft receding horizon control based on Cauthy mutation pigeon inspired optimization (in Chinese)[J]. Scientia Sinica (Technologica), 2018, 48:277–288.

[53] Z. Y. Yang, H. B. Duan, and Y. M. Fan. Unmanned aerial vehicle formation controller design via the behavior mechanism in wild geese based on Levy flight pigeon-inspired optimization (in Chinese) [J]. Scientia Sinica (Technologica), 2018, 48:161–169.

[54] Q. Zhang, and H. Li. MOEA/D: a multi-objective evolutionary algorithm based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6):712–731.

[55] R. Cheng, Y. C. Jin, M. Olhofer, and B. Sendhoff. A Reference Vector Guided Evolutionary Algorithm for Many-objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2016, 20(5):773-791.

[56] Y. Xiang, Y. Zhou, M. Li, and Z. Chen. A Vector Angle-Based Evolutionary Algorithm for Unconstrained Many-Objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(1):131-152.

[57] L.Q. Pan, C. He, Y. Tian, H. D. Wang, X. Y. Zhang, and Y. C. Jin. A classification based surrogate-assisted evolutionary algorithm for expensive many-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2018, 23(1):74-88.

[58] H. K. Chen, Y. Tian, W. Pedrycz, G. H. Wu, R. Wang, and L. Wang. Hyperplane Assisted Evolutionary Algorithm for Many-Objective Optimization Problems[J]. IEEE Transactions on Cybernetics, 2020, 50(7):3367-3380.

[59] Z. M. Hu, Y. Lan, Z. X. Zhang, and X. J. Cai. A many-objective particle swarm optimization algorithm based on multiple criteria for hybrid recommendation system[J]. KSII Transactions on Internet and Information Systems, 2021, 15(2):442-460.

[60] Tian Y, Cheng R, Zhang X, and Jin Y. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization[J]. IEEE Computational Intelligence Magazine, 2017, 12(4):73-87.

[61] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang. Combiningmodel-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion[C]. Proceeding of the 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, 2006: 892–899.

[62] L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(1):29–38.

[63] D. A. Van, V. Gary, and B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis[J]. Evolutionary Computation, 1999, 8(2):125–147.

[64] H. Ishibuchi, H. Masuda, and Y. Nojima. A study on performance evaluation ability of a modified inverted generational distance indicator[C]. Proceeding of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 2015:695–702.

[65] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review[J]. IEEE Transactions on Evolutionary Computation, 2003, 7(2):117–132.

[66] D. Brockhoff, T. Wagner, and H. Trautmann. On the properties of the R2 indicator[C]. Proceeding of the 14th Annual Conference on Genetic and Evolutionary Computation, Philadelphia, Pennsylvania, USA, 2012: 465–472.

[67] J. Bader and E. Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization[J]. Evolutionary Computation, 2011, 19(1):45–76.

[68] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume[J]. European Journal of Operational Research, 2007, 181(3):1653–1669.

[69] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization[J]. Evolutionary Computation, 2007, 15(1):1–28.

[70] Y. Tian, X. Zhang, R. Cheng, and Y. Jin. A multi-objective evolutionary algorithm based on an enhanced inverted generational distance metric[C]. Proceeding of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 2016: 5222–5229.

[71] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin. An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(4):609–622.

[72] 张伟, 刘建昌, 刘圆超, 郑恬子, 杨婉婷. 基于 IGD~+指标的两阶段选择高维多目标进化算法[J/OL]. 控制理论与应用: 1-13 [2022-04-17]. http://kns.cnki.net/kcms/detail/44.1240.TP.20220104.0941.002.html

[73]黎明, 段茹茹, 陈昊, 谢惠华. 基于 IGD~+S 指标的高维多目标进化算法[J]. 模式识别与人工智能, 2019, 32(9):800-810.

[74] E. Zitzler and S. Künzli. Indicator-based Selection in Multiobjective Search[C], Proceeding of the 8th International Conference on Parallel Problem Solving from Nature, Berlin, Heidelberg, 2004: 832-842.

[75] H. Wang, L. Jiao, and X. Yao. Two arch2: An improved two-archive algorithm for many-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2015, 19(4):524–541.

[76]黎明, 段茹茹, 陈昊, 谢惠华. 基于 IGD~+S 指标的高维多目标进化算法[J]. 模式识别与人工智能, 2019, 32(9):800-810.

[77] H. Trautmann, T. Wagner, and D. Brockhoff. R2-EMOA: Focused Multiobjective Search Using R2-Indicator-Based Selection[C]. Proceeding of the International Conference on Learning and Intelligent Optimization, Berlin, Heidelberg, 2013:70–74.

[78] A. D ́ıaz-Manr´ıquez, G. Toscano-Pulido, C. A. C. Coello, and R. Landa-Becerra. A ranking method based on the R2 indicator for manyobjective optimization[C]. Proceeding of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 2013:1523–1530.

[79] R. H. G ́omez and C. A. C. Coello. MOMBI: A new metaheuristic for many-objective optimization based on the R2 indicator[C]. Proceeding of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 2013:2488–2495.

[80] R. H. Gomez and C. A. C. Coello. Improved metaheuristic based on the R2 indicator for many-objective optimization[C]. Proceeding of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 2015:679–686.

[81]李飞, 吴紫恒, 刘阚蓉, 葛二千. 基于 R2 指标和目标空间分解的高维多目标粒子群优化算法[J]. 控制与决策, 2021, 36(9):2085-2094.

[82] Y. Liu, J. Liu, T. Li, Q. Li. An R2 indicator and weight vector-based evolutionary algorithm for multi-objective optimization[J]. Soft Computing, 2020, 24(7):5079-5100.

[83] 王进成, 顾银鲁, 宋桢桢. 基于竞争机制策略的多目标粒子群优化算法[J].宁夏师范学院学报, 2021, 42(10):55-65.

[84] 韩飞, 郑明鹏. 基于三方竞争机制的反向多目标粒子群优化算法[J]. 江苏大学学报(自然科学版), 2021, 42(6):685-693.

[85] 张江江. 高维多目标集成算法研究及应用[D]. 太原:太原科技大学, 2020.

[86] 吴迪. 高维多目标优化算法选择策略研究[D]. 太原:太原科技大学, 2020.

[87] Q. Z. Lin, S. B. Liu, Q. L. Zhu, C. Y. Tang, R. Z. Song, J. Y. Chen, C. A. C. Coello, K. C. Wong, J. Zhang. Particle Swarm Optimization With a Balanceable Fitness Estimation for Many-Objective Optimization Problems[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(1):32-46.

[88] M. Kitani, H. Murakami. One-sample location test based on the sign and Wilcoxon signed-rank tests[J]. Journal of Statistical Computation and Simulation, 2021, 92(3):610-622.