一种基于目标空间转换权重求和的超多目标进化算法

梁正平 [1] 骆婷婷 [1] 王志强 [1, 2] 朱泽轩 [1] 胡凯峰 [2]

摘 要 权重求和是基于分解的超多目标进化算法中常用的方法, 相比其他方法具有计算简单、搜索效率高等优点, 但难以有效处理帕累托前沿面 (Pareto optimal front, PF) 为非凸型的问题. 为充分发挥权重求和方法的优势, 同时又能处理好 PF 为非凸型的问题, 本文提出了一种基于目标空间转换权重求和的超多目标进化算法, 简称 NSGAIII-OSTWS. 该算法的核心是将各种问题的 PF 转换为凸型曲面, 再利用权重求和方法进行优化. 具体地, 首先利用预估 PF 的形状计算个体到预估 PF 的距离; 然后, 根据该距离值将个体映射到目标空间中预估凸型曲面与理想点之间的对应位置; 最后, 采用权重求和函数计算出映射后个体的适应值, 据此实现对问题的进化优化. 为验证 NSGAIII-OSTWS 的有效性, 将 NSGAIII-OSTWS 与 7 个 NSGAIII 的变体, 以及 9 个具有代表性的先进超多目标进化算法在 WFG、DTLZ 和 LSMOP 基准问题上进行对比, 实验结果表明 NSGAIII-OSTWS 具备明显的竞争性能.

关键词 目标空间转换, 权重求和, 超多目标优化, 进化算法

引用格式 梁正平, 骆婷婷, 王志强, 朱泽轩, 胡凯峰. 一种基于目标空间转换权重求和的超多目标进化算法. 自动化学报, 2022, **48**(4): 1060−1078

DOI 10.16383/j.aas.c200483

A Many-objective Evolutionary Algorithm Based on Weighted Sum of Objective Space Transformation

LIANG Zheng-Ping[1] LUO Ting-Ting[1] WANG Zhi-Qiang[1, 2] ZHU Ze-Xuan[1] HU Kai-Feng[2]

Abstract The weighted sum method is a common decomposition method in many-objective evolutionary algorithm based on decomposition. Compared with other methods, it has the advantages of computationally easy and high search efficiency. However, it is difficult for this method to handle the problem with nonconvex Pareto optimal front (PF) effectively. To take full advantage of the weighted sum method and effectively handle the problem with nonconvex PF at the same time, a many-objective evolutionary algorithm based on weighted sum of objective space transformation is proposed, namely NSGAIII-OSTWS. The core of the NSGAIII-OSTWS is to transform the PF of various problems into convex surfaces, and then apply the weighted sum method to optimize the transformed problem. Specifically, the distance between the individual and the estimated PF is calculated firstly. Then all individuals are mapped into the corresponding location between the estimated convex surface and the ideal point according to their distance value. Finally, the fitness values of all mapped individuals are calculated by weighted sum function, and then the evolutionary optimization of the problem is proceeded. In order to verify the effectiveness of NSGAIII-OSTWS, seven variants of NSGAIII, and nine representative advanced many-objective evolutionary algorithms are compared on the WFG, DTLZ and LSMOP benchmark problems. The experimental results show that NSGAIII-OSTWS has obviously competitive performance compared with the comparison algorithms.

Key words Objective space transformation, weighted sum, many-objective optimization, evolutionary algorithm

Citation Liang Zheng-Ping, Luo Ting-Ting, Wang Zhi-Qiang, Zhu Ze-Xuan, Hu Kai-Feng. A many-objective evolutionary algorithm based on weighted sum of objective space transformation. *Acta Automatica Sinica*, 2022, **48**(4): 1060−1078

现实应用中存在各种各样的多目标优化问题 (Multi-objective optimization problems, MOPs), 如电能分配[1]、污水处理控制[2]、服务质量优化[3]、车辆路径规划[4]、软件项目管理[5]、微电网管理[6], 等等. 由于 MOPs 的不同目标函数之间通常相互冲突,

收稿日期 2020-06-30 录用日期 2021-01-26
Manuscript received June 30, 2020; accepted January 26, 2021
国家重点研发计划 (2021YFB2900800), 国家自然科学基金 (61871272), 广东省自然科学基金 (2021A1515011911, 2020A1515010479), 深圳市科技计划 (20200811181752003, GGFW2018020518310863) 资助
Supported by National Key Research and Development Program of China (2021YFB2900800), National Natural Science Foundation of China (61871272), Natural Science Foundation of Guangdong, China (2021A1515011911, 2020A1515010479), Shenzhen Scientific Research and Development Funding Program (20200811181752003, GGFW2018020518310863)
本文责任编委 李成栋
Recommended by Associate Editor LI Cheng-Dong

1. 深圳大学计算机与软件学院 深圳 518060 2. 深圳大学信息中心 深圳 518060
1. College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060 2. Information Center, Shenzhen University, Shenzhen 518060

MOPs 的最优解集不是单一的解而是一组折衷解, 即帕累托最优解集 (Pareto optimal set). 虽然一些性能优越的多目标进化算法 (Multi-objective evolutionary algorithms, MOEAs), 如 NSGA-II[7], SPEA2[8] 和 MOEA-PPF[9] 等能很好的处理 MOPs, 但在解决超多目标优化问题 (Many-objective optimization problems, MaOPs) 时, 上述算法的性能将显著下降. 原因是随着目标函数的增加, 非支配个体在种群中所占的比例将呈指数式增长, 基于帕累托关系的算法会逐渐丧失收敛压力[10]. 为解决该问题, 学术界从不同角度对超多目标进化算法 (Many-objective evolutionary algorithms, MaOEAs) 进行了较广泛的探讨[11−15], 大致可分为以下四类: 1) 基于放松支配. 该类算法的主要思想是通过扩大支配范围来增强种群往 PF 方向收敛的压力. 代表性方法有 ϵ 支配[16], L 支配[17] 和模糊支配[18−19] 等; 2) 基于指标. 该类算法将个体的指标值作为环境选择的衡量标准, 如 IBEA[20], HypE[21] 和 ARMOEA[22] 等; 3) 基于分解. 该类算法将 MaOPs 分解为多个单目标优化问题 (Single-objective optimization problems, SOPs)[23], 随后在进化框架下同时优化这些 SOPs, 如 MOEA/D[24], NSGAIII[25] 和 SPEAR[26] 等. 4) 混合型. 该类算法采用两种或者两种以上的方法来实现对超多目标问题的优化, 如 Two_arch2[27] 和 HpaEA[28] 均融合了基于支配和基于指标的方法, SRA[29] 则采用了多指标的策略. 在上述算法中, 基于分解的 MaOEAs 受到了学术界的高度关注, 该类算法的核心是对参考向量和分解方法的设计.

参考向量主要影响种群在 PF 上分布的均匀性, 即多样性. 在经典的分解类算法 MOEA/D 中, 采用一组均匀分布在目标空间上的向量集作为参考向量. 该参考向量在规则 PF 问题上能取得优越的性能, 但不能很好地处理退化、不连续、凹凸混合等不规则 PF 问题. 原因在于不规则 PF 上参考向量的分布不均匀, 进而造成种群在 PF 上分布不均匀. 针对该问题, 学术界提出了很多对均匀分布的参考向量进行调整的方法, 如 CA-MOEA[30] 通过层次聚类方法来调整每代的参考向量, MOEA-AWA[31] 则根据种群中的精英个体来对参考向量进行调整, 类似的调整方法还有 g-DBEA[32] 和 DDEANS[33] 等. 然而, 由于频繁调整参考向量, 参考向量调整类算法的性能在处理规则 PF 问题时容易恶化[34].

分解方法主要影响种群的搜索效率, 即收敛性. 现有研究表明[35−37], 切比雪夫方法可有效处理各种 PF 形状的问题但搜索效率很低, 权重求和方法虽然不能处理好非凸 PF 问题但搜索效率却很高. 为此, Ishibuchi 等提出了两种改进方案, 分别为自适应切比雪夫和权重求和方法 AS[37], 同时使用切比雪夫和权重求和方法 SS[38], 以综合切比雪夫和权重求和各自的优势. 此外, Wang 等提出了局部权重求和方法 LWS[39], 在综合性能上取得了显著的效果. 但由于 LWS 加入了局部的思想, 一定程度上降低了权重求和方法的搜索效率.

总体而言, 虽然学术界已对基于分解的 MaOEAs 进行了较广泛研究, 但该类方法的性能仍存在较大提升空间. 为尽可能不损害权重求和方法搜索效率高的优势, 同时又能处理好各类 PF 为非凸型的问题, 本文从改进现有分解方法的角度, 提出了一种基于目标空间转换权重求和的超多目标进化算法, 简称 NSGAIII-OSTWS. 其中目标空间转换权重求和 (Objective space transformation based weighted sum, OSTWS) 将各种类型问题的 PF 转换为凸型曲面, 再利用权重求和方法对问题进行优化. 具体地, 首先利用预估 PF 的形状计算个体到预估 PF 的距离; 然后, 根据该距离值将个体映射到目标空间中预估凸型曲面与理想点之间的对应位置; 最后, 采用权重求和函数计算出映射后个体的适应值, 据此实现对问题的优化. 为验证 OSTWS 的有效性, 本文在 NSGAIII 框架的基础上, 将 OSTWS 与现有的 7 个分解方法在 WFG、DTLZ 和 LSMOP 测试问题集上进行了对比, 同时将所提的 NSGAIII-OSTWS 与 9 个具有代表性的 MaOEAs 进行了对比, 实验结果表明 NSGAIII-OSTWS 具备良好的竞争性能.

本文的内容安排如下. 第 1 节介绍与本文相关的背景知识. 第 2 节阐述目标空间转换权重求和方法 OSTWS, 以及基于 OSTWS 的超多目标进化算法 NSGAIII-OSTWS. 第 3 节介绍实验设计、实验结果, 并进行相关讨论. 最后对本文进行总结并指出未来的研究方向.

1　背景知识

1.1　多目标优化问题

一般来说, 一个 MOP 的数学定义可表述为:

$$\begin{aligned} \min \quad & F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \end{aligned} \tag{1}$$

其中, $\boldsymbol{x}$ 是决策空间 Ω 中的 n 维决策向量, m 是目标函数的个数, $\mathbf{R}^m$ 是目标空间. F: $\Omega \to \mathbf{R}^m$ 组成 m 个目标函数. 目标数 m 大于 3 的 MOPs 也被称为超多目标优化问题, 即 MaOPs. 假设 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 是决策空间中的两个不同候选解, $\boldsymbol{x}$ 支配 $\boldsymbol{y}$ (记为 $\boldsymbol{x} \prec \boldsymbol{y}$) 当且仅当 $\forall i \in \{1, 2, \cdots, m\}$, $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})$ 且

$\exists i \in \{1, 2, \cdots, m\}$, $f_i(\boldsymbol{x}) < f_i(\boldsymbol{y})$. 如果候选解 $\boldsymbol{x}$ 不被任何其他解所支配, 则称候选解 $\boldsymbol{x}$ 为帕累托最优解. 所有帕累托最优解的集合称为帕累托最优解集 (Pareto optimal set, PS), 即 $PS = \{\boldsymbol{x} \not\exists\, \boldsymbol{y} \in \Omega, \boldsymbol{x} \succ \boldsymbol{y}\}$. 所有帕累托最优解对应的目标向量构成帕累托最优前沿 (Pareto optimal front, PF), 即 $PF = \{F(\boldsymbol{x}) | \boldsymbol{x} \in PS\}$.

1.2 分解方法

分解方法决定了基于分解的 MaOEAs 的搜索效率, 对算法的性能有着重要影响. 研究者们设计了许多不同的分解方法, 并在各种 MaOPs 上展现出了优越的性能. 常见的分解方法有以下三种:

1) 权重求和 (Weighted sum, WS) 法: WS 方法通过加权的方式将所有目标组合成一个单一的目标, 其数学定义如下:

$$\begin{aligned} & g^{WS}(\boldsymbol{x}|\boldsymbol{w}) = \sum_{i=1}^{m} (\boldsymbol{w}_i f_i(\boldsymbol{x})) \\ & \text{s.t.} \quad \boldsymbol{x} \in \Omega \end{aligned} \tag{2}$$

其中, $\boldsymbol{w} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]$ 为一个参考向量. 在本文中, 满足 $\boldsymbol{w}_i > 0$, 并且 $\sum_{i=1}^{m} (\boldsymbol{w}_i)^2 = 1$. 如图 1(a) 所示, 采用 WS 方法构造的子问题通过在不同的参考向量 $\boldsymbol{w}$ 上进行搜索, 可以快速得到一组近似 PF 点. 在 PF 为凸的情况下, WS 方法能获得一组均匀的 PF 点, 但在 PF 为非凸的情况下, 该方法则会丢失 PF 中的大部分点[24], 从而严重损失种群的多样性, 即 WS 方法不能处理好 PF 为非凸的问题.

2) 切比雪夫 (Techebycheff, TCH) 法: TCH 方法将 MaOP 转化为一个 SOP 的数学定义如下:

$$\begin{aligned} & g^{TCH}(\boldsymbol{x}|\boldsymbol{w}, \boldsymbol{z}^*) = \max_{1 \le i \le m} \left(\frac{1}{\boldsymbol{w}_i} |f_i(\boldsymbol{x}) - \boldsymbol{z}_i^*| \right) \\ & \text{s.t.} \quad \boldsymbol{x} \in \Omega \end{aligned} \tag{3}$$

其中, $\boldsymbol{z}^* = [\boldsymbol{z}_1^*, \boldsymbol{z}_2^*, \cdots, \boldsymbol{z}_m^*]$ 为理想点. 在实际应用中, 如果 $\boldsymbol{w}_i = 0$, 则将 10^{-4} 赋值给 $\boldsymbol{w}_i$, 以此来避免除法的不合理情况. 此外, 为进一步提升所获得解集分布的均匀性, 可将式 (3) 中的 $1/\boldsymbol{w}_i$ 用 $\boldsymbol{w}_i$ 替代[40]. 图 1(b) 为 TCH 方法的二维示意图. 与 WS 方法相比, TCH 方法能处理好任意 PF 形状的 MaOPs. 因此, TCH 方法被广泛应用在各种基于分解的算法中[41−42].

3) 基于惩罚的边界交叉 (Penalty-based boundary intersection, PBI) 法: PBI 方法构造子问题的数学定义如下:

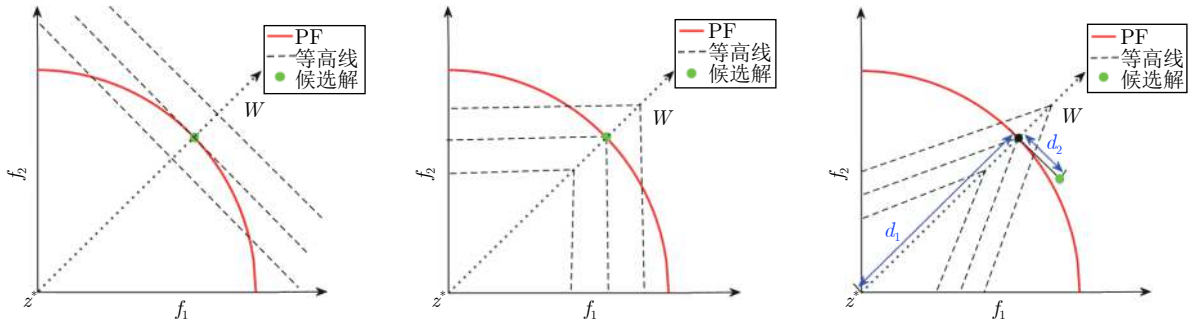
$$\begin{aligned} & g^{PBI}(\boldsymbol{x}|\boldsymbol{w}, \boldsymbol{z}^*) = d_1 + \theta d_2 \\ & \text{s.t.} \quad \boldsymbol{x} \in \Omega \\ & d_1 = \frac{\left\| (F(\boldsymbol{x}) - \boldsymbol{z}^*)\, \boldsymbol{w}^{\mathrm{T}} \right\|}{||\boldsymbol{w}||}, \\ & d_2 = \left\| F(\boldsymbol{x}) - \left(\boldsymbol{z}^* + d_1 \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \right) \right\| \end{aligned} \tag{4}$$

其中, θ ($\theta > 0$) 为一个事先设定的惩罚因子, d_1 为向量 $(F(\boldsymbol{x}) - \boldsymbol{z}^*)$ 在权重向量 $\boldsymbol{w}$ 上的投影长度, d_2 为 $F(\boldsymbol{x})$ 到 $\boldsymbol{w}$ 的垂直距离. 图 1(c) 为 PBI 方法的示意图. 由 PBI 的定义 (式 (4)) 和图 1(c) 可知, θ 是平衡收敛性 (用 d_1 衡量) 和多样性 (用 d_2 衡量) 的关键性参数. 近来有研究表明[43−44], 当 PBI 处理 PF 为凸的问题时, 较大的 θ 可以获得较好的种群分布, 而较小的 θ 值有利于种群更好地收敛到 PF.

2 NSGAIII-OSTWS 算法

2.1 动机

在基于分解的算法中, 分解方法将 MaOP 转化为若干个 SOPs, 然后在进化框架下以协同的方式优化每个 SOPs. 如果没有选择合适的分解方法, 或



(a) WS (b) TCH (c) PBI

图 1 分解方法 WS, TCH 和 PBI 在参考向量 $\boldsymbol{w}$ 上的二维示意图, 其中虚线为等高线

Fig. 1 Illustration of the decomposition methods WS, TCH and PBI on reference vector $\boldsymbol{w}$, where dashed lines are contour lines

者使用的分解方法不能很好地将 MaOP 转化为 SOPs, 基于分解的 MaOEAs 最终获得的种群就有可能无法逼近 PF.

目前, 上节介绍的三种分解方法, 即 WS、TCH 和 PBI, 已在基于分解的 MaOEAs 中被广泛应用. 图 1 是这三种方法在二维目标空间下的示意图. 每个子图中的等高线将目标空间划分为两个区域, 位于同一条等高线上的解具有相同的质量, 靠近理想点 $\boldsymbol{z}^*$ 区域的解质量则要优于另外一区域. 如图 1(a) 所示, WS 的等高线是一条经过候选解且垂直于参考向量的直线, 其优越区域[39] 占整个区域的 1/2. 从图 1(b) 可以看出, TCH 的等高线则为经过候选解和参考向量的两条相互垂直的直线, 其优越区域为 $1/2^m$, 随着目标函数 m 的增加, 该值会显著减小. 图 1(c) 中, PBI 的等高线是经过候选解和参考向量的两条相交直线, 它的优越区域由惩罚因子 θ 决定. θ 值越大, 则优越区域的面积越小, 通常情况下该区域的面积小于整个区域面积的 1/2. 由上述分析可知, WS 的优越区域是最大的. 换言之, 采用 WS 可以更大概率搜索到比目前更优的解, 即收敛速度最快. 然而, 当采用 WS 处理非凸问题时, PF 中的大部分点会被丢失[24], 从而严重损失种群的多样性[39]. 综上所述, 相比 TCH 和 PBI 而言, WS 具有更强的收敛性但却不能处理好非凸问题.

为充分发挥 WS 搜索效率高的优势, 同时又能处理好各类 PF 形状为非凸的问题, 研究者们提出了一些 WS 的改进方法, 如 AS[37] 和 SS[38] 等. 然而, 这些方法的主要思想仅是利用 TCH 方法来处理 WS 不能处理好的凸型 PF 问题, 并未对 WS 方法本身进行改进. 最近, Wang 等[39] 提出了一种新颖的 WS 方法, 即局部权重求和方法 LWS. 该方法的主要思想是对于每个搜索方向, 只在其相邻解中挑选最优解. LWS 方法能够较好处理包括 PF 非凸在内的各类型 PF 问题, 但由于 LWS 在求解最优解时加入了局部的思想, 也在一定程度上降低了 WS 方法搜索效率高的优势. 为验证该结论, 本文将 LWS 方法和 WS 方法分别嵌入到基于分解的算法 NSGAIII[25], 形成算法 NSGAIII-WS 和 NSGAIII-LWS. 图 2 为这两个算法在 PF 为凸的测试问题 ZDT1[45] 上的最终种群分布图. 其中, 运行次数为 20 次, 迭代次数为 120 代, 种群大小为 200, 决策变量数的大小参照文献 [45] 设置, 其它参数与 NSGAIII 保持一致. 从图 2 可以看出, NSGAIII-WS 算法获得的种群具有更好的收敛性. 为尽可能发挥权重求和方法搜索效率高的优势, 同时又能处理好非凸型 PF 问题, 本文提出了一种新颖的方法——目标空间转换权重求和方法.

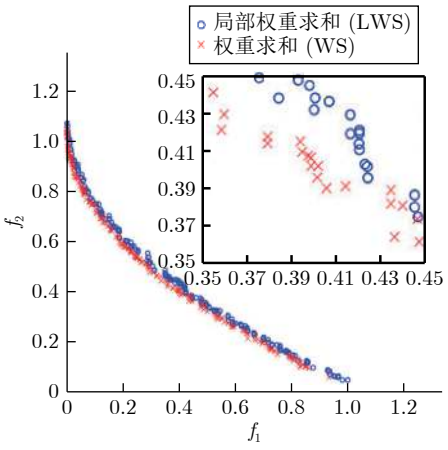


图 2　NSGAIII-WS 和 NSGAIII-LWS 算法在 ZDT1 上获得的最终种群分布

Fig. 2　The final population distribution obtained by NSGAIII-WS and NSGAIII-LWS algorithm on ZDT1

2.2　目标空间转换权重求和方法

目标空间转换权重求和方法 OSTWS 的核心思想, 是将各种问题的 PF 转换为凸型曲面, 并基于该凸型曲面进行求解. OSTWS 方法的伪代码如算法 1 所示.

算法 1. OSTWS 方法

输入. U (种群), N (种群大小), W (参考向量集), C (曲率)

输出. g^{wd} (个体的适应值)

1) 归一化种群 U;

2) 预估出 PF 的形状: p = estimate_PFshape(U);

3) **for** each $\boldsymbol{x} \in U$: /*计算个体转换到凸目标空间中所需的距离值*/

4)　$f_i{}'(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x})}{(\sum_{u=1}^{m}(f_u(\boldsymbol{x}))^p)^{1/p}}, i = 1, \cdots, m;$

5)　$d_1(\boldsymbol{x}) = \sqrt{\sum_{u=1}^{m}(f_u(\boldsymbol{x}))^2};$

6)　$d_2(\boldsymbol{x}) = \sqrt{\sum_{u=1}^{m}(f'_u(\boldsymbol{x}))^2};$

7)　$d(\boldsymbol{x}) = d_1(\boldsymbol{x}) - d_2(\boldsymbol{x})$; /*转换到目标空间中凸曲面与理想点之间的对应位置*/

8)　$f''_i(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x})}{(\sum_{u=1}^{m}(f_u(\boldsymbol{x}))^C)^{1/C}}, i = 1, \cdots, m;$

9)　$\widetilde{f}_i(\boldsymbol{x}) = (1 - d(\boldsymbol{x}))f''_i(\boldsymbol{x})$; /*计算每个个体的适应值*/

10)　**for** each $\boldsymbol{w} \in W$:

11)　　$g^{wd}(\boldsymbol{x}|\boldsymbol{w}) = \sum_{u=1}^{m}(\boldsymbol{w}_u \widetilde{f}_u(\boldsymbol{x}))$;

12)　**end for**

13) **end for**

14) return g^{wd}

首先, 采用 NSGAIII[25] 中的归一化策略对种群 U 进行归一化处理, 然后利用 2REA[46] 中的估 PF 方法预估出 PF 的形状. 种群归一化的目的是维持种群的多样性. 在进化前期, 由于种群中的大部分个体并未收敛到 PF, 归一化会存在较大的误差, 但在进化后期, 大部分个体都已靠近或收敛到 PF, 此时种群归一化所带来的误差会逐渐降低. 预估的过程是先选取一组候选曲率 p 值逐一计算种群中非支配个体到理想点的 L_p 范式值, 并据此计算出各 p 值所对应的标准方差. 方差越小, 代表该 p 值所对应的曲面越能拟合当前种群的分布. 最终选取具有最小方差的 p 值所对应的曲面作为预估的 PF 形状. 之后, 将种群中的所有个体映射到所预估的 PF 上, 映射公式如下:

$$f_i^{'}(\boldsymbol{x})=\frac{f_i(\boldsymbol{x})}{\left(\sum\limits_{u=1}^{m}(f_u(\boldsymbol{x}))^p\right)^{1/p}},\quad i=1,\cdots,m \tag{5}$$

接下来, 以原点作为理想点, 分别计算原始个体与映射个体到理想点之间的欧氏距离值 $d_1(\boldsymbol{x})$ 和 $d_2(\boldsymbol{x})$, 并将 $d_1(\boldsymbol{x})$ 减去 $d_2(\boldsymbol{x})$, 以此得到目标空间转换所需的距离值 $d(\boldsymbol{x})$. 随后, 根据 $d(\boldsymbol{x})$ 值将个体转换到目标空间中凸曲面与理想点之间对应的位置, 从而完成目标空间的转换. 具体的做法是以等距离 $d(\boldsymbol{x})$ 的形式将原始种群中的所有个体映射到预设的凸曲面内. 预设凸曲面的定义如下:

$$\left(\sum_{i=1}^{m}(f_i(\boldsymbol{x}))^C\right)^{1/C}=1,\quad C>0 \tag{6}$$

其中, $[f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x}),\cdots,\ f_m(\boldsymbol{x})]$ 为预设凸曲面上的一个向量, C 为预设凸曲面的曲率值, 种群中个体映射到预设凸曲面内的数学公式如下:

$$f_i^{''}(\boldsymbol{x})=\frac{f_i(\boldsymbol{x})}{\left(\sum\limits_{u=1}^{m}f_u(\boldsymbol{x})^C\right)^{1/C}},\quad i=1,\cdots,m \tag{7}$$

$$\widetilde{f}_i(\boldsymbol{x})=(1-d(\boldsymbol{x}))f_i^{''}(\boldsymbol{x}),\quad i=1,\cdots,m \tag{8}$$

最后, 将最小化求解问题转化为最大化求解问题并采用 WS 方法计算出目标空间转换后个体的适应值, 适应值越大代表个体越优秀.

为更直观地展示 OSTWS 方法, 图 3 示例了使用 OSTWS 方法将线形、凸形和凹形 PF 中的个体转换到凸目标空间的整个过程. 以 3(a) 为例, 首先, 预估出真实 PF 的形状 (直线), 并将种群中的所有个体 (a, b, c) 映射到直线上 (a', b', c'), 然后计算出原始个体 (a, b, c) 与映射后个体 (a', b', c') 之间的欧氏距离值 d_1, d_2, d_3. 值得注意的是, 距离值 (d_1, d_2, d_3) 具有正负之分, 如果原始个体到理想点的欧氏距离大于映射后个体到理想点的欧氏距离, 则上述距离值为正, 否则为负. 接下来, 将原始种群中的所有个体 (a, b, c) 以等距离 (d_1, d_2, d_3) 形式映射到预设凸曲线内, 完成种群到凸目标空间的转换, 转换后的个体为 a'', b'', c''. 最后, 采用 WS 方法逐一计算个体 (a'', b'', c'') 的适应值, 适应值越大的个体被挑选到下一代进化过程中的概率也越大.

2.3 将 OSTWS 方法整合到 NSGAIII 中

在进化算法领域, MOEA/D 和 NSGAII 是两个基于分解的经典算法, NSGAIII 则是 NSGAII 在高维目标空间下的改进版. 相比 MOEA/D, NSGAIII 专门针对 MaOPs, 且在高维空间下能够更好地维持种群的多样性[25, 47−48]. 为公平比较各种分解方法在处理 MaOPs 时的有效性, 本文选择 NSGAIII 作为基础算法, 并将 OSTWS 方法嵌入 NSGAIII 形成新的算法 NSGAIII-OSTWS, 算法 2 为其伪代

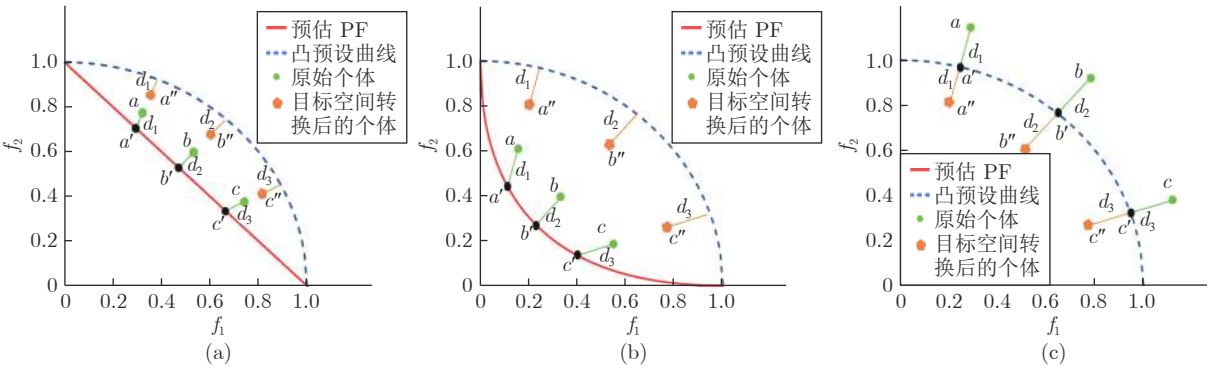


图 3 OSTWS 方法将 PF 形状为线形 (a), 凸形 (b) 和凹形 (c) 种群中的个体转换到凸目标空间的整个过程

Fig. 3 The whole process of transforming the population individuals from linear (a), convex (b) and concave (c) into convex objective space by OSTWS method

码. 首先, 初始化规模为 N 的种群 U 和参考向量 W, 然后对种群 U 进化过程, 直至算法达到最大迭代次数.

算法 2. NSGAIII-OSTWS 算法

输入. N (种群大小), MAXGen (最大迭代次数), C (曲率)

输出. U (种群)

1) 初始化种群 U 和参考向量集 W;
2) **for** $g = 1$ to MAXGen **do**
3)　$S = []$, $U_p = []$, $i = 1$;
4)　Q = OffspringGeneration (U);
5)　$CR = U \bigcup Q$;
6)　(F_1, $F_2, \cdots$) = NondominatedSort(CR);
7)　**repeat**
8)　$S = S \bigcup F_i$ and $i = i+1$;
9)　**until** $|S| \geq N$
10)　**if** $|S| == N$
11)　$U_p = S$;
12)　**else**
13)　$U_p = \cup_{j=1}^{i-1} F_j$;
14)　g^{wd} = OSTWS(S, N, W, C);
15)　根据最大值 g^{wd} 将种群 S 中的个体归属至 $\boldsymbol{w}$ ($w \in W$), 并统计前 F_{i-1} 层中个体归属至每个 $\boldsymbol{w}$ 上的个体数量;
16)　通过 $\boldsymbol{w}$ 归属个体的数量和 g^{wd} 值从 F_i 层中选取出 $N - |U_p|$ 个个体加入至 U_p 中;
17)　**end if**
18)　$U = U_p$;
19) **end for**
20) return U

在每一代进化过程中, 根据上一代种群进行父代选择 (二进制锦标赛[25])、个体交叉 (二进制交叉[49]) 和变异 (多项式变异[50]), 生成规模为 N 的子代种群 Q. 之后, 合并种群 Q 和 U 形成规模为 $2N$ 的种群 CR 并对种群 CR 进行非支配层 (F_1, F_2 等) 排序[7]. 接下来将位于前 $i-1$ 层的个体直接作为下一代种群的个体, 其中 i 为满足 $\sum_{j=1}^{i} F_j \geq N$ 的最小整数. 若 $\sum_{j=1}^{i} F_j > N$, 则再调用 OSTWS 方法从 F_i 层中挑选 $N - \sum_{j=1}^{i-1} F_j$ 个个体进入下一代种群. 具体地, 首先采用 OSTWS 方法计算出 S 中前 i 层中的个体到所有参考向量的权重求和值 (g^{wd}), 然后识别出具有最大 g^{wd} 值的个体的参考向量, 并将该个体归属至该参考向量上; 接下来, 统计前 F_{i-1} 层中个体归属至每个 W 上的个体数量; 最后, 在归属数量最少的参考向量上选取出 F_i 中具有最大 g^{wd} 值的个体并将该其加入至 U_p 中, 重复该步骤直至 $|U_p| = N$, 最后输出种群 U.

2.4　时间复杂度分析

时间复杂度是衡量算法性能的一个重要方面, 影响算法的整体计算开销. 下面根据 NSGAIII-OSTWS 的主要流程对其时间复杂度进行详细分析. 算法 2 第 4 行的子代个体生成中, 二进制交叉和多项式变异需要 O(DN) 的计算开销, 其中 D 为决策变量的数量. 算法 2 第 6 行对规模为 $2N$ 的种群进行非支配层排序要花费 O($N \log^{m-2} N$)[25] 的计算代价. 算法 2 第 15 行, 即算法 1, 其计算开销包括以下四方面: 1) 将规模为 $2N$ 的种群进行归一化 (算法 1 第 1 行), 需要 O(mN) 的计算开销. 2) 计算出 $2N$ 个个体到预设曲面的距离 (算法 1 第 4~7 行), 需要 O(mN) 的计算开销. 3) 将 $2N$ 个个体的目标空间转换到特定曲面内 (算法 1 第 8~9 行), 需要 O(mN) 的计算开销. 4) 计算目标空间转换后 $2N$ 个个体到参考向量 W 的权重求和值 (算法 1 第 10~12 行), 需要 O($mN|W|$) 的计算开销, 其中 $|W|$ 是参考向量的个数. 因此, 算法 1 的时间复杂度为 O($mN|W|$). 之后, 在最坏情况下 NSGAIII-OSTWS 需对 $2N$ 个个体进行参考向量的归属, 此时需要花费 O($mN|W|$) 的计算代价. 最后, NSGAIII-OSTWS 需挑选 $N - |U_p|$ 个个体至下一代进化过程需要 O($L|W|$) 的计算量, 其中 $L = |F_l|$. 此外, 在通常情况下, $N \approx |W|$, $N > m$. 考虑到以上因素和计算结果, NSGAIII-OSTWS 的时间复杂度为 Max(O(DN), O($mN|W|$)).

3　实验与分析

3.1　实验设置

1) 测试问题和评价指标

为检验 NSGAIII-OSTWS 的性能, 本文选取了超多目标优化领域中使用最为广泛的两组测试问题集 WFG[51]、DTLZ[52] 和最新的大规模决策变量测试问题集 LSMOP[53]. 在 DTLZ 中, DTLZ8-9 为带约束的问题, 因此本文只考虑对 DTLZ1-7 问题的研究. 参照文献 [39], WFG 和 DTLZ 中的决策变量数目统一设置为 $D = 100$, 其中 WFG 中的位置变量数设为 $k = m - 1$. LSMOP 的相关参数与原文[53] 保持一致.

为定量评估算法的求解性能, 分别采用世代距离 (Generational distance, GD[54]), 覆盖 PF(Coverage over the pareto front, CPF[55]) 和修正的反转

世代距离 (Modified inverted generational distance, IGD$^+$[56]) 进行收敛性、多样性和综合性 (平衡收敛性和多样性的能力) 性能衡量, 其计算公式分别如下:

$$d_{GD} = \frac{1}{|U|}\sum_{i=1}^{|U|}\min_{a_j\in A,\, u_i\in U} d(u_i, a_j) \tag{9}$$

$$d_{IGD^+} = \frac{1}{|A|}\sum_{j=1}^{|A|}\min_{u_i\in U,\, a_j\in A}\sqrt{\sum_{k=1}^{m}\left(\max\{u_i{}^k - a_j^k, 0\}\right)^2} \tag{10}$$

其中 U 为算法输出的最终种群, A 为测试问题 PF 面上的一组均匀参考点. 对于目标维度不相同的缩放问题 (如: WFG), 在计算 GD 和 IGD$^+$指标之前需对 A 和 U 进行归一化处理[48]. GD 和 IGD$^+$值越小, 代表算法的性能越好. 参照文献 [57], 本实验设置计算 GD 和 IGD$^+$所需的参考点数为 10 000.

2) 参数设置

由于分解算法的种群规模取决于参数 H 和 m, 其中 H 为沿每个目标轴所考虑的分区. 为公平起见, 本章所测试的目标维度及其对应的种群大小统一设置为表 1 所示. 为避免参考向量分布不均匀的情况, 当 $m > 5$ 时, 本文采用两层参考向量生成方法生成参考向量[58]. 交叉和变异操作所需的参数设置如表 2 所示. 在 NSGAIII-OSTWS 中, 曲率 C 值的参数经验设置为 2, 其影响在实验分析部分进行研究. 比较算法的其他参数设置与其原始论文保持一致. 所有算法的终止准则被指定为最大迭代次数 (MAXGen), 本文中所有测试问题的 MAXGen 设为 300. 每个算法在每个测试问题上独立运行 20 次. 为检验算法性能的显著性差异, 采用 Wilcoxon[59] 秩和检验来评估一种算法在 GD 或 IGD$^+$值方面是否优于另一种算法. 符号 "+", "−" 和 "≈"表示相应的竞争算法在 5% 的显著性水平上分别比所提算法 NSGAIII-OSTWS 更好, 更差和无统计性差异.

表 1 种群大小设置

Table 1 Setting of the population size

目标数 (m)	分割数 (H)	种群大小 (N)
3	12	91
5	6	210
8	3, 2	156
10	3, 2	275

3.2 实验结果与分析

1) OSTWS 方法的有效性验证

为验证目标空间转换权重求和方法 (OSTWS) 的有效性, 本小节将 OSTWS 方法与其他 7 个分解方法, 即切比雪夫方法 (TCH)、惩罚边界交叉方法 (PBI[24])、局部权重求和方法 (LWS[39])、自适应切比雪夫和权重求和方法 (AS[37])、同时使用切比雪夫和权重求和方法 (SS[38])、自适应 Lp 方法 (PaS[36]) 和自适应惩罚方法 (APS[60]) 在算法 NSGAIII 的框架上进行比较实验. 表 3、表 4 和表 5 分别统计了上述分解方法在 DTLZ1-DTLZ7、WFG1-WFG9 和 LSMOP1-LSMOP9 测试问题上所获得的 GD 均值和标准差 (括号内为标准差), 其中每个问题的最佳结果以灰色背景突出显示. 图 4为各个算法在所有测试问题上的平均 IGD$^+$表现分, 分值越小表示该算法整体性能越好.

表 2 交叉变异参数设置

Table 2 Parameter settings for crossover and mutation

参数名	参数值
交叉概率 (P_c)	1.0
变异概率 (P_m)	1/ D
交叉分布指标 (η_c)	20
变异分布指标 (η_m)	20

由表 3、表 4 和表 5 可以看出, NSGAIII-OSTWS 在绝大部分 DTLZ 测试问题上都取得了最佳 GD 均值, 此外, 虽然 NSGAIII-OSTWS 没能在所有的 WFG 和 LSMOP 测试问题中获得最具竞争力的 GD 性能, 但总体上获得了最优的性能, 这表明本文所提出的 OSTWS 方法是非常有效的, 原因在于充分利用了 WS 方法搜索效率高的优势.

下面具体分析各个算法在 DTLZ1-DTLZ7、WFG1-WFG9 和 LSMOP1-LSMOP9 测试问题上的性能. DTLZ1 是一个线性问题, NSGAIII-OSTWS 在该问题上获得了最佳的 GD 值, 即在线性的 DTLZ1 问题上, OSTWS 方法的收敛性要强于其它所对比的分解方法. DTLZ2-4 和 WFG4-9 为凹问题, 对于 DTLZ2-4, NSGAIII-OSTWS 获得了最优性能. 为直观展示各个算法在凹问题上的收敛性能, 图 5 例举了所有算法在 10 维 DTLZ2问题上的最终种群分布图. 从图 5 可以看出, 算法 NSGAIII-OSTWS 的种群分布在目标空间 [0, 1] 内, 其它算法的种群大部分都分布在目标空间 [0, 1.2] 内, 这表明算法 NSGAIII-OSTWS 能很好的将种群收敛到 PF 上, 而其它算法却不能. 虽然 WFG4-9 同样为凹问题, 但相对于 DTLZ2-4 来说, WFG4-9 测试问题具有多峰、带欺骗和变量不可分离等更加复杂的特点, 从而给算法带来了更大的挑战. 但从表 4 可以看出, NSGAIII-OSTWS 在大部分 WFG4-9 测试问题上都能取得最佳的性能指标值. DTLZ5-

6 和 WFG3 为退化问题, 对于 DTLZ5-6, NSGAIII-OSTWS 仍然表现出了最佳的收敛能力. 在 WFG3 上, 由 GD 指标的统计数据可知 NSGAIII-OSTWS 的性能则一般. WFG1 是一个凹凸混合问题, WFG2 和 DTLZ7 为不连续问题, 在这三个问题上, NSGAIII-OSTWS 的性能会有所下降. 原因是 OSTWS 很难快速且准确地预估出此类 PF 的形状, 需要消耗一定的迭代次数, 从而降低了种群的收敛速度. 但本文所采用的预估 PF 方法最终能较好地预估出这些问题的 PF 形状, 故在处理这些不规则测试问题时依然取得了较好的性能. 相对于 DTLZ 和 WFG 测试问题集, LSMOP 具有更大规模的决策变量, 种群的收敛难度增大, 对现有的超多目标算法提出了更大的挑战. 由于 OSTWS 继承了权重求和分解方法的高搜索效率, 因此相对于原始的 NSGAIII 算法, NSGAIII-OSTWS 在难收敛的大规

表 3　OSTWS, LWS, TCH, PBI, AS, SS, PaS 和 APS 方法在框架为 NSGAIII, 测试问题为 DTLZ1-7 上获得的 GD 值统计结果 (均值和标准差). 每个实例算法中的最好结果以加粗突出显示

Table 3　The statistical results (mean and standard deviation) of the GD values obtained by OSTWS, LWS, TCH, PBI, AS, SS, PaS and APS methods on the NSGAIII framework and DTLZ1-7 test problems. The best average value among the algorithms for each instance is highlighted in bold

Problem	m	NSGAIII-OSTWS	NSGAIII-LWS	NSGAIII-TCH	NSGAIII-PBI	NSGAIII-AS	NSGAIII- SS	NSGAIII-PaS	NSGAIII-APS
DTLZ1	3	**6.915×10^{1}**	7.734×10^{1}	7.411×10^{1}	7.140×10^{1}	7.678×10^{1}	7.749×10^{1}	7.554×10^{1}	7.376×10^{1}
		(1.2×10^{1})	$(9.4\times10^{0})-$	$(9.3\times10^{0})\approx$	$(1.1\times10^{1})\approx$	$(1.1\times10^{1})-$	$(7.3\times10^{0})-$	$(9.4\times10^{0})\approx$	$(9.8\times10^{0})\approx$
	5	**4.218×10^{1}**	7.562×10^{1}	8.102×10^{1}	7.015×10^{1}	7.870×10^{1}	1.294×10^{2}	7.993×10^{1}	6.507×10^{1}
		(4.5×10^{0})	$(6.6\times10^{0})-$	$(8.9\times10^{0})-$	$(7.6\times10^{0})-$	$(8.1\times10^{0})-$	$(9.7\times10^{0})-$	$(8.0\times10^{0})-$	$(7.6\times10^{0})-$
	8	**4.881×10^{1}**	9.120×10^{1}	7.814×10^{1}	9.002×10^{1}	7.875×10^{1}	2.458×10^{2}	7.411×10^{1}	8.732×10^{1}
		(1.2×10^{1})	$(7.3\times10^{0})-$	$(1.1\times10^{1})-$	$(1.1\times10^{1})-$	$(1.2\times10^{1})-$	$(5.8\times10^{1})-$	$(9.0\times10^{0})-$	$(8.8\times10^{0})-$
	10	**4.422×10^{1}**	9.338×10^{1}	7.014×10^{1}	7.334×10^{1}	6.757×10^{1}	2.672×10^{2}	7.299×10^{1}	7.538×10^{1}
		(1.8×10^{1})	$(4.9\times10^{0})-$	$(8.2\times10^{0})-$	$(2.4\times10^{1})-$	$(5.3\times10^{0})-$	$(5.1\times10^{1})-$	$(7.3\times10^{0})-$	$(3.2\times10^{1})-$
DTLZ2	3	**1.681×10^{-3}**	3.971×10^{-3}	3.465×10^{-3}	4.582×10^{-3}	3.552×10^{-3}	6.753×10^{-3}	3.585×10^{-3}	4.529×10^{-3}
		(2.3×10^{-4})	$(6.1\times10^{-4})-$	$(3.9\times10^{-4})-$	$(6.8\times10^{-4})-$	$(4.2\times10^{-4})-$	$(1.1\times10^{-3})-$	$(5.2\times10^{-4})-$	$(6.4\times10^{-4})-$
	5	**3.337×10^{-3}**	4.526×10^{-3}	4.968×10^{-3}	6.740×10^{-3}	4.996×10^{-3}	1.003×10^{-2}	4.903×10^{-3}	6.756×10^{-3}
		(9.7×10^{-5})	$(3.6\times10^{-4})-$	$(3.0\times10^{-4})-$	$(5.3\times10^{-4})-$	$(4.3\times10^{-4})-$	$(1.4\times10^{-3})-$	$(4.1\times10^{-4})-$	$(4.0\times10^{-4})-$
	8	**1.058×10^{-2}**	1.265×10^{-2}	1.439×10^{-2}	2.362×10^{-2}	1.470×10^{-2}	7.435×10^{-2}	1.516×10^{-2}	2.488×10^{-2}
		(3.3×10^{-4})	$(2.5\times10^{-3})-$	$(1.7\times10^{-3})-$	$(4.6\times10^{-3})-$	$(1.4\times10^{-3})-$	$(3.5\times10^{-2})-$	$(2.5\times10^{-3})-$	$(3.2\times10^{-3})-$
	10	**1.070×10^{-2}**	1.503×10^{-2}	1.130×10^{-2}	1.733×10^{-2}	1.133×10^{-2}	7.939×10^{-2}	1.227×10^{-2}	1.996×10^{-2}
		(4.2×10^{-3})	$(6.0\times10^{-3})-$	$(2.7\times10^{-3})-$	$(7.3\times10^{-3})-$	$(1.8\times10^{-3})-$	$(5.4\times10^{-2})-$	$(3.5\times10^{-3})-$	$(6.6\times10^{-3})-$
DTLZ3	3	8.888×10^{1}	8.622×10^{1}	8.323×10^{1}	**8.322×10^{1}**	8.246×10^{1}	8.259×10^{1}	8.910×10^{1}	8.705×10^{1}
		(1.2×10^{1})	$(1.5\times10^{1})\approx$	$(1.5\times10^{1})\approx$	**$(9.7\times10^{0})\approx$**	$(1.1\times10^{1})\approx$	$(6.9\times10^{0})\approx$	$(1.3\times10^{1})\approx$	$(1.2\times10^{1})\approx$
	5	**6.174×10^{1}**	9.024×10^{1}	8.343×10^{1}	9.531×10^{1}	8.414×10^{1}	1.255×10^{2}	8.277×10^{1}	9.465×10^{1}
		(7.3×10^{0})	$(9.1\times10^{0})-$	$(1.0\times10^{1})-$	$(1.3\times10^{1})-$	$(1.1\times10^{1})-$	$(9.7\times10^{0})-$	$(8.8\times10^{0})-$	$(9.7\times10^{0})-$
	8	**8.605×10^{1}**	1.266×10^{2}	1.220×10^{2}	1.536×10^{2}	1.176×10^{2}	3.001×10^{2}	1.293×10^{2}	1.434×10^{2}
		(2.0×10^{1})	$(1.4\times10^{1})-$	$(9.6\times10^{0})-$	$(2.3\times10^{1})-$	$(9.2\times10^{0})-$	$(8.2\times10^{1})-$	$(1.3\times10^{1})-$	$(2.5\times10^{1})-$
	10	**7.830×10^{1}**	1.340×10^{2}	1.201×10^{2}	1.427×10^{2}	1.157×10^{2}	3.629×10^{2}	1.273×10^{2}	1.314×10^{2}
		(3.5×10^{1})	$(2.3\times10^{1})-$	$(1.4\times10^{1})-$	$(4.5\times10^{1})-$	$(7.3\times10^{0})-$	$(7.1\times10^{1})-$	$(2.7\times10^{1})-$	$(2.9\times10^{1})-$
DTLZ4	3	**1.918×10^{-3}**	4.026×10^{-3}	3.588×10^{-3}	4.068×10^{-3}	3.455×10^{-3}	7.379×10^{-3}	3.242×10^{-3}	4.393×10^{-3}
		(3.1×10^{-4})	$(1.1\times10^{-3})-$	$(1.1\times10^{-3})-$	$(2.1\times10^{-3})-$	$(1.1\times10^{-3})-$	$(3.0\times10^{-3})-$	$(1.3\times10^{-3})-$	$(1.6\times10^{-3})-$
	5	**3.506×10^{-3}**	5.160×10^{-3}	5.502×10^{-3}	8.623×10^{-3}	5.326×10^{-3}	7.816×10^{-3}	5.367×10^{-3}	8.775×10^{-3}
		(5.0×10^{-4})	$(5.4\times10^{-4})-$	$(4.4\times10^{-4})-$	$(1.4\times10^{-3})-$	$(2.9\times10^{-4})-$	$(2.3\times10^{-3})-$	$(3.5\times10^{-4})-$	$(1.0\times10^{-3})-$
	8	**1.658×10^{-2}**	2.169×10^{-2}	2.858×10^{-2}	3.486×10^{-2}	2.597×10^{-2}	7.637×10^{-2}	1.887×10^{-2}	3.592×10^{-2}
		(2.0×10^{-2})	$(1.7\times10^{-2})\approx$	$(1.8\times10^{-2})-$	$(1.6\times10^{-2})-$	$(1.7\times10^{-2})-$	$(3.6\times10^{-2})-$	$(5.4\times10^{-3})-$	$(2.2\times10^{-2})-$
	10	**7.670×10^{-3}**	1.737×10^{-2}	1.282×10^{-2}	2.060×10^{-2}	1.336×10^{-2}	1.052×10^{-1}	1.047×10^{-2}	1.792×10^{-2}
		(2.0×10^{-3})	$(1.8\times10^{-2})\approx$	$(7.1\times10^{-3})-$	$(1.2\times10^{-2})-$	$(8.2\times10^{-3})-$	$(6.0\times10^{-2})-$	$(4.2\times10^{-3})-$	$(4.8\times10^{-3})-$

表 3 OSTWS, LWS, TCH, PBI, AS, SS, PaS 和 APS 方法在框架为 NSGAIII, 测试问题为 DTLZ1-7 上获得的 GD 值统计结果 (均值和标准差). 每个实例算法中的最好结果以加粗突出显示 (续表)

Table 3 The statistical results (mean and standard deviation) of the GD values obtained by OSTWS, LWS, TCH, PBI, AS, SS, PaS and APS methods on the NSGAIII framework and DTLZ1-7 test problems. The best average value among the algorithms for each instance is highlighted in bold (continued table)

Problem	m	NSGAIII-OSTWS	NSGAIII-LWS	NSGAIII-TCH	NSGAIII-PBI	NSGAIII-AS	NSGAIII- SS	NSGAIII-PaS	NSGAIII-APS
DTLZ5	3	4.566×10^{-3}	$\mathbf{4.388\times10^{-3}}$	4.698×10^{-3}	5.107×10^{-3}	4.836×10^{-3}	5.209×10^{-3}	5.353×10^{-3}	5.055×10^{-3}
		(7.2×10^{-4})	$\mathbf{(7.7\times10^{-4})\approx}$	$(6.3\times10^{-4})\approx$	$(7.8\times10^{-4})-$	$(7.4\times10^{-4})\approx$	$(8.2\times10^{-4})-$	$(8.9\times10^{-4})-$	$(5.4\times10^{-4})-$
	5	$\mathbf{6.411\times10^{-2}}$	4.279×10^{-1}	1.027×10^{-1}	6.492×10^{-2}	1.194×10^{-1}	2.345×10^{-1}	1.186×10^{-1}	7.556×10^{-2}
		$\mathbf{(1.7\times10^{-2})}$	$(8.2\times10^{-2})-$	$(1.6\times10^{-2})-$	$(1.5\times10^{-2})\approx$	$(2.3\times10^{-2})-$	$(3.6\times10^{-2})-$	$(1.4\times10^{-2})-$	$(1.6\times10^{-2})-$
	8	$\mathbf{2.795\times10^{-1}}$	5.134×10^{-1}	4.167×10^{-1}	4.142×10^{-1}	5.253×10^{-1}	1.082×10^{0}	5.319×10^{-1}	4.527×10^{-1}
		$\mathbf{(5.2\times10^{-2})}$	$(1.1\times10^{-1})-$	$(7.0\times10^{-2})-$	$(5.7\times10^{-2})-$	$(8.8\times10^{-2})-$	$(5.9\times10^{-1})-$	$(1.1\times10^{-1})-$	$(9.5\times10^{-2})-$
	10	$\mathbf{3.845\times10^{-1}}$	1.266×10^{0}	1.283×10^{0}	8.411×10^{-1}	1.660×10^{0}	2.054×10^{0}	1.668×10^{0}	9.914×10^{-1}
		$\mathbf{(2.3\times10^{-1})}$	$(3.6\times10^{-1})-$	$(3.1\times10^{-1})-$	$(2.7\times10^{-1})-$	$(2.1\times10^{-1})-$	$(6.5\times10^{-1})-$	$(2.3\times10^{-1})-$	$(2.1\times10^{-1})-$
DTLZ6	3	$\mathbf{3.555\times10^{0}}$	4.484×10^{0}	4.150×10^{0}	4.294×10^{0}	4.055×10^{0}	6.531×10^{0}	4.099×10^{0}	4.164×10^{0}
		$\mathbf{(3.4\times10^{-1})}$	$(3.6\times10^{-1})-$	$(4.0\times10^{-1})-$	$(4.3\times10^{-1})-$	$(2.3\times10^{-1})-$	$(2.6\times10^{-1})-$	$(4.0\times10^{-1})-$	$(4.2\times10^{-1})-$
	5	$\mathbf{2.454\times10^{0}}$	1.135×10^{1}	8.566×10^{0}	6.659×10^{0}	8.595×10^{0}	7.759×10^{0}	8.606×10^{0}	6.597×10^{0}
		$\mathbf{(2.8\times10^{-1})}$	$(2.3\times10^{-1})-$	$(5.4\times10^{-1})-$	$(1.8\times10^{-1})-$	$(3.0\times10^{-1})-$	$(4.1\times10^{-1})-$	$(3.6\times10^{-1})-$	$(2.7\times10^{-1})-$
	8	$\mathbf{1.235\times10^{1}}$	2.182×10^{1}	1.927×10^{1}	2.201×10^{1}	1.915×10^{1}	2.548×10^{1}	1.933×10^{1}	2.174×10^{1}
		$\mathbf{(8.7\times10^{-1})}$	$(2.3\times10^{0})-$	$(8.5\times10^{-1})-$	$(3.1\times10^{0})-$	$(9.4\times10^{-1})-$	$(6.2\times10^{0})-$	$(1.1\times10^{0})-$	$(4.0\times10^{0})-$
	10	$\mathbf{1.344\times10^{1}}$	2.871×10^{1}	2.511×10^{1}	2.395×10^{1}	2.535×10^{1}	2.887×10^{1}	2.501×10^{1}	2.203×10^{1}
		$\mathbf{(1.6\times10^{0})}$	$(8.3\times10^{0})-$	$(1.6\times10^{0})-$	$(1.0\times10^{1})-$	$(1.1\times10^{0})-$	$(1.1\times10^{1})-$	$(4.0\times10^{0})-$	$(8.9\times10^{0})-$
DTLZ7	3	$\mathbf{1.385\times10^{-2}}$	1.479×10^{-2}	1.476×10^{-2}	1.788×10^{-2}	1.538×10^{-2}	1.780×10^{-2}	1.628×10^{-2}	1.839×10^{-2}
		$\mathbf{(2.3\times10^{-3})}$	$(2.5\times10^{-3})\approx$	$(2.0\times10^{-3})\approx$	$(2.3\times10^{-3})-$	$(1.9\times10^{-3})-$	$(3.3\times10^{-3})-$	$(3.7\times10^{-3})-$	$(2.5\times10^{-3})-$
	5	$\mathbf{8.419\times10^{-3}}$	9.474×10^{-3}	9.755×10^{-3}	1.481×10^{-2}	9.498×10^{-3}	1.712×10^{-2}	9.146×10^{-3}	1.534×10^{-2}
		$\mathbf{(1.2\times10^{-3})}$	$(1.4\times10^{-3})-$	$(1.0\times10^{-3})-$	$(1.6\times10^{-3})-$	$(1.0\times10^{-3})-$	$(1.6\times10^{-3})-$	$(1.3\times10^{-3})\approx$	$(1.4\times10^{-3})-$
	8	$\mathbf{2.742\times10^{-2}}$	2.987×10^{-2}	3.999×10^{-2}	3.594×10^{-2}	3.700×10^{-2}	5.275×10^{-2}	4.093×10^{-2}	3.597×10^{-2}
		$\mathbf{(1.8\times10^{-3})}$	$(4.4\times10^{-3})\approx$	$(3.2\times10^{-3})-$	$(5.5\times10^{-3})-$	$(5.2\times10^{-3})-$	$(5.6\times10^{-3})-$	$(5.4\times10^{-3})-$	$(4.7\times10^{-3})-$
	10	2.928×10^{-2}	$\mathbf{2.449\times10^{-2}}$	2.800×10^{-2}	2.893×10^{-2}	3.052×10^{-2}	4.273×10^{-2}	3.055×10^{-2}	2.869×10^{-2}
		(2.3×10^{-3})	$\mathbf{(3.6\times10^{-3})+}$	$(2.5\times10^{-3})\approx$	$(2.0\times10^{-3})\approx$	$(1.9\times10^{-3})\approx$	$(3.5\times10^{-3})-$	$(3.1\times10^{-3})\approx$	$(3.6\times10^{-3})\approx$
$+/-/\approx$			1/21/6	0/23/5	0/24/4	0/25/3	0/27/1	0/25/3	0/24/4

表 4 OSTWS, LWS, TCH, PBI, AS, SS, PaS 和 APS 方法在框架为 NSGAIII, 测试问题集为 WFG1-9 上获得的 GD 值统计结果 (均值和标准差). 每个实例算法中的最好结果以加粗突出显示

Table 4 The statistical results (mean and standard deviation) of the GD values obtained by OSTWS, LWS, TCH, PBI, AS, SS, PaS and APS methods on the NSGAIII framework and WFG1-9 test problems. The best average value among the algorithms for each instance is highlighted in bold

Problem	m	NSGAIII-OSTWS	NSGAIII-LWS	NSGAIII-TCH	NSGAIII-PBI	NSGAIII-AS	NSGAIII- SS	NSGAIII-PaS	NSGAIII-APS
WFG1	3	$\mathbf{4.082\times10^{-2}}$	4.125×10^{-2}	4.435×10^{-2}	4.357×10^{-2}	4.475×10^{-2}	4.206×10^{-2}	4.454×10^{-2}	4.368×10^{-2}
		$\mathbf{(6.0\times10^{-4})}$	$(9.1\times10^{-4})\approx$	$(8.9\times10^{-4}-$	$(7.2\times10^{-4})-$	$(7.5\times10^{-4})-$	$(1.1\times10^{-3})-$	$(1.2\times10^{-3})-$	$(6.7\times10^{-4})-$
	5	2.789×10^{-2}	$\mathbf{2.670\times10^{-2}}$	3.220×10^{-2}	2.936×10^{-2}	3.194×10^{-2}	2.790×10^{-2}	3.225×10^{-2}	2.960×10^{-2}
		(9.2×10^{-4})	$\mathbf{(4.3\times10^{-4})+}$	$(6.2\times10^{-4})-$	$(3.9\times10^{-4})-$	$(7.8\times10^{-4})-$	$(7.8\times10^{-4})-$	$(6.2\times10^{-4})-$	$(3.0\times10^{-4})-$
	8	$\mathbf{3.323\times10^{-2}}$	3.429×10^{-2}	3.472×10^{-2}	3.483×10^{-2}	3.504×10^{-2}	3.624×10^{-2}	3.506×10^{-2}	3.446×10^{-2}
		$\mathbf{(9.2\times10^{-4})}$	$(1.3\times10^{-3})-$	$(8.6\times10^{-4})-$	$(9.1\times10^{-4})-$	$(1.3\times10^{-3})-$	$(3.4\times10^{-3})-$	$(1.5\times10^{-3})-$	$(1.4\times10^{-3})-$
	10	$\mathbf{2.474\times10^{-2}}$	2.585×10^{-2}	2.589×10^{-2}	2.614×10^{-2}	2.546×10^{-2}	2.816×10^{-2}	2.535×10^{-2}	2.607×10^{-2}
		$\mathbf{(5.6\times10^{-4})}$	$(5.3\times10^{-4})-$	$(1.2\times10^{-3})-$	$(9.3\times10^{-4})-$	$(9.5\times10^{-4})-$	$(1.6\times10^{-3})-$	$(9.3\times10^{-4})-$	$(8.2\times10^{-4})-$
WFG2	3	5.354×10^{-3}	$\mathbf{5.103\times10^{-3}}$	5.846×10^{-3}	6.124×10^{-3}	5.965×10^{-3}	8.529×10^{-3}	6.085×10^{-3}	6.088×10^{-3}
		(6.5×10^{-4})	$\mathbf{(4.8\times10^{-4})\approx}$	$(6.4\times10^{-4})-$	$(4.6\times10^{-4})-$	$(4.8\times10^{-4})-$	$(1.6\times10^{-3})-$	$(7.1\times10^{-4})-$	$(4.9\times10^{-4})-$

表 4　OSTWS, LWS, TCH, PBI, AS, SS, PaS 和 APS 方法在框架为 NSGAIII, 测试问题集为 WFG1-9 上获得的 GD 值统计结果 (均值和标准差). 每个实例算法中的最好结果以加粗突出显示 (续表)

Table 4　The statistical results (mean and standard deviation) of the GD values obtained by OSTWS, LWS, TCH, PBI, AS, SS, PaS and APS methods on the NSGAIII framework and WFG1-9 test problems. The best average value among the algorithms for each instance is highlighted in bold (continued table)

Problem	m	NSGAIII-OSTWS	NSGAIII-LWS	NSGAIII-TCH	NSGAIII-PBI	NSGAIII-AS	NSGAIII- SS	NSGAIII-PaS	NSGAIII-APS
	5	**4.885×10^{-3}**	5.663×10^{-3}	7.042×10^{-3}	5.902×10^{-3}	7.329×10^{-3}	6.705×10^{-3}	6.924×10^{-3}	6.009×10^{-3}
		(2.2×10^{-4})	$(6.0\times10^{-4})-$	$(6.3\times10^{-4})-$	$(2.1\times10^{-4})-$	$(5.3\times10^{-4})-$	$(8.3\times10^{-4})-$	$(1.1\times10^{-3})-$	$(2.1\times10^{-4})-$
	8	**8.277×10^{-3}**	1.011×10^{-2}	9.969×10^{-3}	9.745×10^{-3}	1.025×10^{-2}	1.269×10^{-2}	1.006×10^{-2}	1.018×10^{-2}
		(7.0×10^{-4})	$(1.1\times10^{-3})-$	$(5.7\times10^{-4})-$	$(1.6\times10^{-3})-$	$(1.0\times10^{-3})-$	$(2.8\times10^{-3})-$	$(9.9\times10^{-4})-$	$(3.1\times10^{-3})-$
	10	1.528×10^{-2}	1.447×10^{-2}	1.309×10^{-2}	1.329×10^{-2}	**1.142×10^{-2}**	1.460×10^{-2}	1.179×10^{-2}	1.317×10^{-2}
		(1.8×10^{-3})	$(1.8\times10^{-3})\approx$	$(1.9\times10^{-3})+$	$(2.3\times10^{-3})+$	**$(1.6\times10^{-3})+$**	$(2.7\times10^{-3})\approx$	$(1.1\times10^{-3})+$	$(2.6\times10^{-3})+$
WFG3	3	**1.154×10^{-2}**	1.273×10^{-2}	1.480×10^{-2}	1.684×10^{-2}	1.4610×10^{-2}	2.620×10^{-2}	1.497×10^{-2}	1.693×10^{-2}
		(1.5×10^{-3})	$(1.6\times10^{-3})-$	$(1.2\times10^{-3})-$	$(2.0\times10^{-3})-$	$(1.1\times10^{-3})-$	$(2.0\times10^{-3})-$	$(2.0\times10^{-3})-$	$(1.1\times10^{-3})-$
	5	3.912×10^{-2}	**3.555×10^{-2}**	1.130×10^{-1}	8.826×10^{-2}	1.320×10^{-1}	5.550×10^{-2}	1.156×10^{-1}	7.736×10^{-2}
		(4.2×10^{-3})	**$(4.5\times10^{-3})+$**	$(1.1\times10^{-2})-$	$(2.7\times10^{-2})-$	$(1.2\times10^{-2})-$	$(7.5\times10^{-3})-$	$(1.5\times10^{-2})-$	$(2.0\times10^{-2})-$
	8	6.263×10^{-1}	8.328×10^{-1}	6.008×10^{-1}	**5.867×10^{-1}**	7.676×10^{-1}	5.991×10^{-1}	6.118×10^{-1}	6.314×10^{-1}
		(1.3×10^{-1})	$(8.3\times10^{-2})-$	$(1.2\times10^{-1})\approx$	**$(1.2\times10^{-1})\approx$**	$(2.5\times10^{-1})-$	$(9.2\times10^{-2})\approx$	$(1.2\times10^{-1})\approx$	$(2.3\times10^{-1})\approx$
	10	2.374×10^{0}	3.674×10^{0}	2.840×10^{0}	**1.902×10^{0}**	3.110×10^{0}	2.402×10^{0}	3.252×10^{0}	1.965×10^{0}
		(8.6×10^{-1})	$(6.4\times10^{-1})-$	$(1.0\times10^{0})-$	**$(6.0\times10^{-1})\approx$**	$(9.7\times10^{-1})-$	$(2.7\times10^{-1})-$	$(1.1\times10^{0})-$	$(6.7\times10^{-1})\approx$
WFG4	3	**1.387×10^{-3}**	2.231×10^{-3}	3.001×10^{-3}	3.370×10^{-3}	2.893×10^{-3}	4.412×10^{-3}	2.953×10^{-3}	3.284×10^{-3}
		(1.1×10^{-4})	$(1.5\times10^{-4})-$	$(2.5\times10^{-4})-$	$(2.1\times10^{-4})-$	$(2.3\times10^{-4})-$	$(3.6\times10^{-4})-$	$(1.5\times10^{-4})-$	$(2.4\times10^{-4})-$
	5	3.717×10^{-3}	**2.834×10^{-3}**	5.696×10^{-3}	4.746×10^{-3}	5.847×10^{-3}	4.401×10^{-3}	5.766×10^{-3}	4.725×10^{-3}
		(6.3×10^{-5})	**$(7.6\times10^{-5})+$**	$(3.1\times10^{-4})-$	$(8.3\times10^{-5})-$	$(3.6\times10^{-4})-$	$(1.4\times10^{-4})-$	$(3.4\times10^{-4})-$	$(6.5\times10^{-5})-$
	8	1.263×10^{-2}	**1.244×10^{-2}**	1.543×10^{-2}	1.501×10^{-2}	1.497×10^{-2}	1.462×10^{-2}	1.517×10^{-2}	1.437×10^{-2}
		(3.8×10^{-4})	**$(3.6\times10^{-4})\approx$**	$(6.9\times10^{-4})-$	$(7.1\times10^{-4})-$	$(6.3\times10^{-4})-$	$(1.4\times10^{-3})-$	$(5.4\times10^{-4})-$	$(1.5\times10^{-3})-$
	10	**7.624×10^{-3}**	1.344×10^{-2}	1.203×10^{-2}	8.833×10^{-3}	1.210×10^{-2}	1.401×10^{-2}	1.211×10^{-2}	8.060×10^{-3}
		(8.9×10^{-4})	$(5.8\times10^{-4})-$	$(1.4\times10^{-4})-$	$(1.8\times10^{-3})-$	$(2.0\times10^{-4})-$	$(6.9\times10^{-4})-$	$(2.1\times10^{-4})-$	$(1.3\times10^{-3})\approx$
WFG5	3	**2.770×10^{-3}**	3.404×10^{-3}	3.979×10^{-3}	4.138×10^{-3}	3.927×10^{-3}	5.689×10^{-3}	3.861×10^{-3}	4.077×10^{-3}
		(7.4×10^{-5})	$(2.0\times10^{-4})-$	$(2.2\times10^{-4})-$	$(1.7\times10^{-4})-$	$(1.8\times10^{-4})-$	$(5.0\times10^{-4})-$	$(1.9\times10^{-4})-$	$(1.6\times10^{-4})-$
	5	4.094×10^{-3}	**3.377×10^{-3}**	6.612×10^{-3}	4.771×10^{-3}	6.671×10^{-3}	4.338×10^{-3}	6.681×10^{-3}	4.755×10^{-3}
		(8.3×10^{-5})	**$(5.6\times10^{-5})+$**	$(4.5\times10^{-4})-$	$(8.2\times10^{-5})-$	$(4.7\times10^{-4})-$	$(1.9\times10^{-4})-$	$(5.1\times10^{-4})-$	$(8.3\times10^{-5})-$
	8	1.288×10^{-2}	**1.281×10^{-2}**	1.747×10^{-2}	1.543×10^{-2}	1.767×10^{-2}	1.385×10^{-2}	1.737×10^{-2}	1.548×10^{-2}
		(2.2×10^{-4})	**$(5.9\times10^{-4})\approx$**	$(5.6\times10^{-4})-$	$(2.0\times10^{-4})-$	$(5.8\times10^{-4})-$	$(1.1\times10^{-3})-$	$(4.6\times10^{-4})-$	$(3.0\times10^{-4})-$
	10	9.292×10^{-3}	1.366×10^{-2}	1.149×10^{-2}	**8.550×10^{-3}**	1.158×10^{-2}	1.246×10^{-2}	1.159×10^{-2}	8.604×10^{-3}
		(3.8×10^{-4})	$(3.0\times10^{-4})-$	$(3.3\times10^{-4})-$	**$(3.8\times10^{-4})+$**	$(3.0\times10^{-4})-$	$(8.8\times10^{-4})-$	$(3.2\times10^{-4})-$	$(4.0\times10^{-4})+$
WFG6	3	**2.151×10^{-3}**	3.052×10^{-3}	3.902×10^{-3}	4.170×10^{-3}	3.933×10^{-3}	5.274×10^{-3}	3.913×10^{-3}	4.134×10^{-3}
		(1.6×10^{-4})	$(1.9\times10^{-4})-$	$(2.0\times10^{-4})-$	$(3.0\times10^{-4})-$	$(2.7\times10^{-4})-$	$(4.4\times10^{-4})-$	$(2.4\times10^{-4})-$	$(2.3\times10^{-4})-$
	5	3.999×10^{-3}	**3.168×10^{-3}**	8.235×10^{-3}	4.969×10^{-3}	8.134×10^{-3}	4.872×10^{-3}	7.986×10^{-3}	4.941×10^{-3}
		(9.9×10^{-5})	**$(7.5\times10^{-5})+$**	$(9.9\times10^{-4})-$	$(1.2\times10^{-4})-$	$(8.0\times10^{-4})-$	$(2.0\times10^{-4})-$	$(8.7\times10^{-4})-$	$(1.1\times10^{-4})-$
	8	1.250×10^{-2}	**1.215×10^{-2}**	1.800×10^{-2}	1.544×10^{-2}	1.799×10^{-2}	1.536×10^{-2}	1.7820×10^{-2}	1.555×10^{-2}
		(2.4×10^{-4})	**$(7.6\times10^{-4})\approx$**	$(5.8\times10^{-4})-$	$(2.3\times10^{-4})-$	$(7.9\times10^{-4})-$	$(9.2\times10^{-4})-$	$(9.1\times10^{-4})-$	$(3.0\times10^{-4})-$
	10	**7.483×10^{-3}**	1.238×10^{-2}	1.230×10^{-2}	7.812×10^{-3}	1.241×10^{-2}	1.463×10^{-2}	1.244×10^{-2}	8.120×10^{-3}
		(5.2×10^{-4})	$(6.1\times10^{-4})-$	$(4.0\times10^{-4})-$	$(3.6\times10^{-4})-$	$(3.1\times10^{-4})-$	$(6.7\times10^{-4})-$	$(3.2\times10^{-4})-$	$(9.1\times10^{-4})-$
WFG7	3	**8.882×10^{-4}**	2.105×10^{-3}	3.270×10^{-3}	5.073×10^{-3}	3.280×10^{-3}	7.455×10^{-3}	3.182×10^{-3}	4.750×10^{-3}
		(4.9×10^{-5})	$(3.2\times10^{-4})-$	$(7.6\times10^{-4})-$	$(7.9\times10^{-4})-$	$(3.7\times10^{-4})-$	$(3.0\times10^{-3})-$	$(4.5\times10^{-4})-$	$(5.8\times10^{-4})-$

表 4 OSTWS, LWS, TCH, PBI, AS, SS, PaS 和 APS 方法在框架为 NSGAIII, 测试问题集为 WFG1-9 上获得的 GD 值统计结果 (均值和标准差). 每个实例算法中的最好结果以加粗突出显示 (续表)

Table 4 The statistical results (mean and standard deviation) of the GD values obtained by OSTWS, LWS, TCH, PBI, AS, SS, PaS and APS methods on the NSGAIII framework and WFG1-9 test problems. The best average value among the algorithms for each instance is highlighted in bold (continued table)

Problem	m	NSGAIII-OSTWS	NSGAIII-LWS	NSGAIII-TCH	NSGAIII-PBI	NSGAIII-AS	NSGAIII- SS	NSGAIII-PaS	NSGAIII-APS
WFG8	5	3.323×10^{-3}	$\mathbf{2.647\times10^{-3}}$	7.618×10^{-3}	5.695×10^{-3}	8.228×10^{-3}	4.176×10^{-3}	8.303×10^{-3}	5.888×10^{-3}
		(8.2×10^{-5})	$\mathbf{(9.6\times10^{-5})+}$	$(1.6\times10^{-3})-$	$(5.7\times10^{-4})-$	$(1.8\times10^{-3})-$	$(3.2\times10^{-4})-$	$(1.6\times10^{-3})-$	$(7.7\times10^{-4})-$
	8	$\mathbf{1.168\times10^{-2}}$	1.245×10^{-2}	1.742×10^{-2}	1.599×10^{-2}	1.745×10^{-2}	1.330×10^{-2}	1.736×10^{-2}	1.606×10^{-2}
		$\mathbf{(6.5\times10^{-4})}$	$(2.2\times10^{-3})-$	$(6.5\times10^{-4})-$	$(6.3\times10^{-4})-$	$(5.7\times10^{-4})-$	$(1.0\times10^{-3})-$	$(6.3\times10^{-4})-$	$(8.2\times10^{-4})-$
	10	$\mathbf{6.602\times10^{-3}}$	1.129×10^{-2}	1.217×10^{-2}	8.449×10^{-3}	1.222×10^{-2}	1.308×10^{-2}	1.220×10^{-2}	8.566×10^{-3}
		$\mathbf{(6.2\times10^{-4})}$	$(8.8\times10^{-4})-$	$(3.1\times10^{-4})-$	$(5.0\times10^{-4})-$	$(2.6\times10^{-4})-$	$(5.0\times10^{-4})-$	$(3.6\times10^{-4})-$	$(7.1\times10^{-4})-$
	3	$\mathbf{4.390\times10^{-3}}$	5.139×10^{-3}	5.619×10^{-3}	5.561×10^{-3}	5.792×10^{-3}	7.016×10^{-3}	5.703×10^{-3}	5.550×10^{-3}
		$\mathbf{(2.7\times10^{-4})}$	$(2.4\times10^{-4})-$	$(2.6\times10^{-4})-$	$(2.5\times10^{-4})-$	$(2.5\times10^{-4})-$	$(4.8\times10^{-4})-$	$(3.5\times10^{-4})-$	$(3.0\times10^{-4})-$
	5	4.796×10^{-3}	$\mathbf{4.301\times10^{-3}}$	7.939×10^{-3}	5.018×10^{-3}	7.786×10^{-3}	5.403×10^{-3}	7.969×10^{-3}	5.053×10^{-3}
		(1.1×10^{-4})	$\mathbf{(1.0\times10^{-4})+}$	$(4.1\times10^{-4})-$	$(1.0\times10^{-4})-$	$(4.8\times10^{-4})-$	$(3.6\times10^{-4})-$	$(5.9\times10^{-4})-$	$(9.2\times10^{-5})-$
	8	$\mathbf{1.307\times10^{-2}}$	1.308×10^{-2}	1.832×10^{-2}	1.569×10^{-2}	1.834×10^{-2}	1.524×10^{-2}	1.828×10^{-2}	1.560×10^{-2}
		$\mathbf{(3.0\times10^{-4})}$	$(1.1\times10^{-3})\approx$	$(5.5\times10^{-4})-$	$(2.5\times10^{-4})-$	$(5.0\times10^{-4})-$	$(1.1\times10^{-3})-$	$(4.7\times10^{-4})-$	$(6.4\times10^{-4})-$
	10	9.844×10^{-3}	1.405×10^{-2}	1.227×10^{-2}	$\mathbf{9.342\times10^{-3}}$	1.229×10^{-2}	1.346×10^{-2}	1.243×10^{-2}	9.774×10^{-3}
		(2.6×10^{-4})	$(4.3\times10^{-4})-$	$(4.0\times10^{-4})-$	$\mathbf{(4.8\times10^{-4})+}$	$(3.0\times10^{-4})-$	$(8.9\times10^{-4})-$	$(4.6\times10^{-4})-$	$(8.4\times10^{-4})\approx$
WFG9	3	$\mathbf{2.200\times10^{-3}}$	6.110×10^{-3}	5.287×10^{-3}	8.679×10^{-3}	5.360×10^{-3}	8.726×10^{-3}	4.796×10^{-3}	8.011×10^{-3}
		$\mathbf{(2.8\times10^{-4})}$	$(6.8\times10^{-4})-$	$(5.0\times10^{-4})-$	$(1.1\times10^{-3})-$	$(6.0\times10^{-4})-$	$(9.6\times10^{-4})-$	$(7.7\times10^{-4})-$	$(6.3\times10^{-4})-$
	5	$\mathbf{4.144\times10^{-3}}$	4.250×10^{-3}	8.592×10^{-3}	7.838×10^{-3}	8.773×10^{-3}	5.496×10^{-3}	8.772×10^{-3}	7.720×10^{-3}
		$\mathbf{(1.1\times10^{-4})}$	$(2.1\times10^{-4})-$	$(1.3\times10^{-3})-$	$(3.6\times10^{-4})-$	$(2.0\times10^{-3})-$	$(4.4\times10^{-4})-$	$(1.3\times10^{-3})-$	$(4.5\times10^{-4})-$
	8	$\mathbf{1.339\times10^{-2}}$	1.570×10^{-2}	1.810×10^{-2}	1.814×10^{-2}	1.827×10^{-2}	1.508×10^{-2}	1.801×10^{-2}	1.793×10^{-2}
		$\mathbf{(3.2\times10^{-4})}$	$(1.8\times10^{-3})-$	$(7.5\times10^{-4})-$	$(5.5\times10^{-4})-$	$(6.2\times10^{-4})-$	$(1.6\times10^{-3})-$	$(7.3\times10^{-4})-$	$(4.3\times10^{-4})-$
	10	$\mathbf{1.082\times10^{-2}}$	1.533×10^{-2}	1.245×10^{-2}	1.288×10^{-2}	1.280×10^{-2}	1.315×10^{-2}	1.276×10^{-2}	1.274×10^{-2}
		$\mathbf{(3.5\times10^{-4})}$	$(3.1\times10^{-4})-$	$(3.2\times10^{-4})-$	$(4.1\times10^{-4})-$	$(4.6\times10^{-4})-$	$(8.6\times10^{-4})-$	$(4.3\times10^{-4})-$	$(3.8\times10^{-4})-$
$+/-/\approx$			7/22/7	1/33/2	3/31/2	1/35/0	0/32/4	1/34/1	2/30/4

表 5 OSTWS, LWS, TCH, PBI, AS, SS, PaS 和 APS 方法在框架为 NSGAIII, 测试问题集为 LSMOP1-9 上获得的 GD 值统计结果 (均值和标准差). 每个实例算法中的最好结果以加粗突出显示

Table 5 The statistical results (mean and standard deviation) of the GD values obtained by OSTWS, LWS, TCH, PBI, AS, SS, PaS and APS methods on the NSGAIII framework and LSMOP1-9 test problems. The best average value among the algorithms for each instance is highlighted in bold

Problem	m	NSGAIII-OSTWS	NSGAIII-LWS	NSGAIII-TCH	NSGAIII-PBI	NSGAIII-AS	NSGAIII-SS	NSGAIII-PaS	NSGAIII-APS
LSMOP1	3	$\mathbf{8.526\times10^{-1}}$	9.451×10^{-1}	8.776×10^{-1}	9.691×10^{-1}	8.904×10^{-1}	1.439×10^{0}	9.296×10^{-1}	9.070×10^{-1}
		$\mathbf{(1.3\times10^{-1})}$	$(9.7\times10^{-2})-$	$(1.1\times10^{-1})\approx$	$(1.4\times10^{-1})-$	$(1.4\times10^{-1})\approx$	$(7.8\times10^{-1})\approx$	$(1.4\times10^{-1})\approx$	$(1.3\times10^{-1})\approx$
	5	4.829×10^{-1}	5.596×10^{-1}	5.858×10^{-1}	5.483×10^{-1}	5.800×10^{-1}	8.027×10^{-1}	$\mathbf{4.784\times10^{-1}}$	5.518×10^{-1}
		(2.1×10^{-1})	$(5.7\times10^{-2})-$	$(5.3\times10^{-2})-$	$(9.3\times10^{-2})-$	$(4.8\times10^{-2})-$	$(8.0\times10^{-2})-$	$\mathbf{(1.0\times10^{-1})\approx}$	$(4.5\times10^{-2})-$
	8	$\mathbf{5.409\times10^{-1}}$	8.855×10^{-1}	8.447×10^{-1}	8.037×10^{-1}	9.307×10^{-1}	1.343×10^{0}	7.875×10^{-1}	8.568×10^{-1}
		$\mathbf{(1.4\times10^{-1})}$	$(5.9\times10^{-2})-$	$(1.6\times10^{-1})-$	$(2.0\times10^{-1})-$	$(2.1\times10^{-1})-$	$(1.7\times10^{-1})-$	$(1.1\times10^{-1})-$	$(1.4\times10^{-1})-$
	10	$\mathbf{4.598\times10^{-1}}$	8.812×10^{-1}	8.967×10^{-1}	6.762×10^{-1}	9.021×10^{-1}	9.689×10^{-1}	6.232×10^{-1}	9.011×10^{-1}
		$\mathbf{(1.1\times10^{-1})}$	$(1.01\times10^{-1})-$	$(7.9\times10^{-2})-$	$(1.4\times10^{-1})-$	$(8.4\times10^{-2})-$	$(1.1\times10^{-1})-$	$(7.5\times10^{-2})-$	$(1.0\times10^{-1})-$
LSMOP2	3	$\mathbf{7.646\times10^{-3}}$	1.026×10^{-2}	9.405×10^{-3}	9.825×10^{-3}	9.403×10^{-3}	1.105×10^{-2}	9.698×10^{-3}	9.490×10^{-3}
		$\mathbf{(1.8\times10^{-4})}$	$(2.3\times10^{-4})-$	$(1.3\times10^{-4})-$	$(1.6\times10^{-4})-$	$(1.4\times10^{-4})-$	$(1.1\times10^{-3})-$	$(1.7\times10^{-4})-$	$(1.8\times10^{-4})-$
	5	$\mathbf{6.197\times10^{-3}}$	8.530×10^{-3}	7.660×10^{-3}	7.866×10^{-3}	7.661×10^{-3}	9.447×10^{-3}	7.920×10^{-3}	7.623×10^{-3}
		$\mathbf{(8.9\times10^{-5})}$	$(9.0\times10^{-5})-$	$(7.0\times10^{-5})-$	$(5.8\times10^{-5})-$	$(1.6\times10^{-4})-$	$(5.4\times10^{-4})-$	$(5.0\times10^{-5})-$	$(4.5\times10^{-5})-$
	8	$\mathbf{1.243\times10^{-2}}$	2.217×10^{-2}	1.890×10^{-2}	1.948×10^{-2}	1.839×10^{-2}	1.792×10^{-2}	1.982×10^{-2}	1.824×10^{-2}
		$\mathbf{(6.0\times10^{-4})}$	$(2.6\times10^{-3})-$	$(2.2\times10^{-3})-$	$(6.8\times10^{-4})-$	$(1.5\times10^{-3})-$	$(3.7\times10^{-3})-$	$(1.5\times10^{-3})-$	$(2.1\times10^{-3})-$
	10	$\mathbf{9.467\times10^{-3}}$	1.364×10^{-2}	1.230×10^{-2}	1.271×10^{-2}	1.248×10^{-2}	1.658×10^{-2}	1.213×10^{-2}	1.239×10^{-2}
		$\mathbf{(2.1\times10^{-4})}$	$(6.5\times10^{-4})-$	$(1.3\times10^{-3})-$	$(1.6\times10^{-4})-$	$(1.1\times10^{-3})-$	$(4.0\times10^{-3})-$	$(3.3\times10^{-4})-$	$(1.3\times10^{-3})-$

表 5　OSTWS, LWS, TCH, PBI, AS, SS, PaS 和 APS 方法在框架为 NSGAIII, 测试问题集为 LSMOP1-9 上获得的 GD 值统计结果 (均值和标准差). 每个实例算法中的最好结果以加粗突出显示 (续表)

Table 5　The statistical results (mean and standard deviation) of the GD values obtained by OSTWS, LWS, TCH, PBI, AS, SS, PaS and APS methods on the NSGAIII framework and LSMOP1-9 test problems. The best average value among the algorithms for each instance is highlighted in bold (continued table)

Problem	m	NSGAIII-OSTWS	NSGAIII-LWS	NSGAIII-TCH	NSGAIII-PBI	NSGAIII-AS	NSGAIII-SS	NSGAIII-PaS	NSGAIII-APS
LSMOP3	3	2.990×10^{2}	$\mathbf{2.763\times10^{2}}$	2.993×10^{2}	3.366×10^{2}	3.316×10^{2}	4.780×10^{2}	3.243×10^{2}	3.793×10^{2}
		(1.32×10^{2})	$\mathbf{(7.0\times10^{1})\approx}$	$(8.8\times10^{1})\approx$	$(1.8\times10^{2})\approx$	$(8.7\times10^{1})\approx$	$(2.5\times10^{2})-$	$(1.3\times10^{2})\approx$	$(1.0\times10^{2})-$
	5	$\mathbf{6.116\times10^{2}}$	8.207×10^{2}	9.284×10^{2}	9.599×10^{2}	9.630×10^{2}	1.906×10^{3}	9.785×10^{2}	9.834×10^{2}
		$\mathbf{(2.8\times10^{2})}$	$(1.7\times10^{2})-$	$(1.5\times10^{2})-$	$(5.5\times10^{2})-$	$(2.2\times10^{2})-$	$(4.6\times10^{2})-$	$(4.95\times10^{2})-$	$(2.7\times10^{2})-$
	8	$\mathbf{1.369\times10^{3}}$	2.207×10^{3}	2.607×10^{3}	1.927×10^{3}	3.040×10^{3}	3.418×10^{3}	2.167×10^{3}	3.087×10^{3}
		$\mathbf{(5.8\times10^{2})}$	$(5.8\times10^{2})-$	$(6.1\times10^{2})-$	$(6.2\times10^{2})-$	$(5.2\times10^{2})-$	$(8.0\times10^{2})-$	$(8.0\times10^{2})-$	$(8.9\times10^{2})-$
	10	3.379×10^{3}	3.133×10^{3}	4.097×10^{3}	$\mathbf{2.945\times10^{3}}$	4.655×10^{3}	3.510×10^{3}	3.373×10^{3}	4.029×10^{3}
		(1.7×10^{3})	$(1.1\times10^{3})\approx$	$(9.2\times10^{2})\approx$	$\mathbf{(1.5\times10^{3})\approx}$	$(1.2\times10^{3})-$	$(7.9\times10^{2})\approx$	$(1.9\times10^{3})\approx$	$(9.6\times10^{2})\approx$
LSMOP4	3	$\mathbf{3.073\times10^{-2}}$	3.744×10^{-2}	3.730×10^{-2}	4.045×10^{-2}	3.704×10^{-2}	4.004×10^{-2}	4.034×10^{-2}	3.742×10^{-2}
		$\mathbf{(2.0\times10^{-3})}$	$(9.4\times10^{-4})-$	$(1.4\times10^{-3})-$	$(1.2\times10^{-3})-$	$(1.1\times10^{-3})-$	$(4.9\times10^{-3})-$	$(1.0\times10^{-3})-$	$(1.2\times10^{-3})-$
	5	$\mathbf{2.598\times10^{-2}}$	3.942×10^{-2}	3.656×10^{-2}	3.888×10^{-2}	3.547×10^{-2}	3.750×10^{-2}	4.093×10^{-2}	3.641×10^{-2}
		$\mathbf{(1.8\times10^{-3})}$	$(1.8\times10^{-3})-$	$(2.1\times10^{-3})-$	$(2.0\times10^{-3})-$	$(1.5\times10^{-3})-$	$(4.6\times10^{-3})-$	$(1.6\times10^{-3})-$	$(1.6\times10^{-3})-$
	8	$\mathbf{1.995\times10^{-2}}$	3.397×10^{-2}	2.209×10^{-2}	3.297×10^{-2}	2.324×10^{-2}	5.114×10^{-2}	3.301×10^{-2}	2.257×10^{-2}
		$\mathbf{(7.8\times10^{-4})}$	$(1.3\times10^{-3})-$	$(3.5\times10^{-3})-$	$(1.4\times10^{-3})-$	$(4.2\times10^{-3})-$	$(9.6\times10^{-3})-$	$(1.4\times10^{-3})-$	$(4.2\times10^{-3})-$
	10	$\mathbf{1.459\times10^{-2}}$	2.521×10^{-2}	1.490×10^{-2}	2.260×10^{-2}	1.548×10^{-2}	2.585×10^{-2}	2.283×10^{-2}	1.613×10^{-2}
		$\mathbf{(4.8\times10^{-4})}$	$(1.3\times10^{-3})-$	$(8.0\times10^{-4})\approx$	$(7.0\times10^{-4})-$	$(1.1\times10^{-3})-$	$(6.9\times10^{-3})-$	$(1.0\times10^{-3})-$	$(2.4\times10^{-3})-$
LSMOP5	3	2.605×10^{0}	2.694×10^{0}	2.534×10^{0}	2.780×10^{0}	2.516×10^{0}	3.782×10^{0}	2.772×10^{0}	$\mathbf{2.292\times10^{0}}$
		(3.9×10^{-1})	$(3.2\times10^{-1})\approx$	$(3.2\times10^{-1})\approx$	$(2.7\times10^{-1})\approx$	$(4.6\times10^{-1})\approx$	$(4.9\times10^{-1})-$	$(4.1\times10^{-1})\approx$	$\mathbf{(2.4\times10^{-1})+}$
	5	$\mathbf{2.314\times10^{0}}$	3.210×10^{0}	3.370×10^{0}	3.510×10^{0}	3.075×10^{0}	5.678×10^{0}	3.810×10^{0}	3.079×10^{0}
		$\mathbf{(1.4\times10^{0})}$	$(2.3\times10^{-1})-$	$(3.1\times10^{-1})-$	$(8.2\times10^{-1})-$	$(2.4\times10^{-1})-$	$(5.9\times10^{-1})-$	$(4.8\times10^{-1})-$	$(3.1\times10^{-1})-$
	8	$\mathbf{4.708\times10^{0}}$	7.771×10^{0}	7.613×10^{0}	5.93×10^{0}	7.218×10^{0}	7.384×10^{0}	8.816×10^{0}	7.327×10^{0}
		$\mathbf{(2.2\times10^{0})}$	$(1.1\times10^{0})-$	$(9.5\times10^{-1})-$	$(3.6\times10^{0})\approx$	$(9.6\times10^{-1})-$	$(1.7\times10^{0})-$	$(1.8\times10^{0})-$	$(1.1\times10^{0})-$
	10	5.077×10^{0}	6.883×10^{0}	6.279×10^{0}	6.707×10^{0}	6.522×10^{0}	$\mathbf{4.854\times10^{0}}$	7.631×10^{0}	6.118×10^{0}
		(7.7×10^{-1})	$(6.8\times10^{-1})-$	$(9.4\times10^{-1})-$	$(2.0\times10^{0})-$	$(9.3\times10^{-1})-$	$\mathbf{(6.3\times10^{-1})\approx}$	$(2.8\times10^{-1})-$	$(7.2\times10^{-1})-$
LSMOP6	3	4.402×10^{3}	3.127×10^{3}	$\mathbf{2.938\times10^{3}}$	3.477×10^{3}	3.410×10^{3}	4.088×10^{3}	4.089×10^{3}	3.025×10^{3}
		(2.5×10^{3})	$(1.4\times10^{3})\approx$	$\mathbf{(1.1\times10^{3})\approx}$	$(1.7\times10^{3})\approx$	$(1.6\times10^{3})\approx$	$(2.0\times10^{3})\approx$	$(2.5\times10^{3})\approx$	$(2.0\times10^{3})+$
	5	5.002×10^{3}	5.108×10^{3}	5.354×10^{3}	4.366×10^{3}	5.637×10^{3}	1.202×10^{4}	$\mathbf{3.587\times10^{3}}$	5.220×10^{3}
		(2.6×10^{3})	$(1.3\times10^{3})\approx$	$(1.0\times10^{3})\approx$	$(2.5\times10^{3})\approx$	$(1.4\times10^{3})\approx$	$(3.4\times10^{3})-$	$\mathbf{(1.2\times10^{3})+}$	$(1.6\times10^{3})\approx$
	8	2.206×10^{4}	$\mathbf{2.041\times10^{4}}$	2.484×10^{4}	3.608×10^{4}	2.475×10^{4}	7.647×10^{4}	3.385×10^{4}	2.425×10^{4}
		(4.9×10^{3})	$\mathbf{(5.6\times10^{3})\approx}$	$(9.6\times10^{3})\approx$	$(9.8\times10^{3})-$	$(8.7\times10^{3})\approx$	$(1.8\times10^{4})-$	$(1.3\times10^{4})-$	$(5.4\times10^{3})\approx$
	10	1.933×10^{4}	$\mathbf{1.924\times10^{4}}$	2.847×10^{4}	3.601×10^{4}	2.661×10^{4}	5.022×10^{4}	3.410×10^{4}	2.844×10^{4}
		(4.8×10^{3})	$\mathbf{(2.7\times10^{3})\approx}$	$(6.9\times10^{3})-$	$(8.8\times10^{3})-$	$(9.5\times10^{3})-$	$(8.7\times10^{3})-$	$(7.8\times10^{3})-$	$(6.2\times10^{3})-$
LSMOP7	3	$\mathbf{5.540\times10^{2}}$	8.520×10^{2}	8.707×10^{2}	9.200×10^{2}	8.946×10^{2}	2.598×10^{3}	9.381×10^{2}	1.013×10^{3}
		$\mathbf{(1.4\times10^{2})}$	$(4.2\times10^{2})-$	$(3.8\times10^{2})-$	$(4.0\times10^{2})-$	$(2.5\times10^{2})-$	$(8.7\times10^{2})-$	$(2.8\times10^{2})-$	$(3.0\times10^{2})-$
	5	4.597×10^{3}	4.424×10^{3}	5.261×10^{3}	$\mathbf{4.238\times10^{3}}$	5.665×10^{3}	1.403×10^{4}	4.530×10^{3}	5.627×10^{3}
		(1.6×10^{3})	$(1.8\times10^{3})\approx$	$(2.1\times10^{3})\approx$	$\mathbf{(9.4\times10^{2})\approx}$	$(1.6\times10^{3})-$	$(3.9\times10^{3})-$	$(1.8\times10^{3})\approx$	$(3.4\times10^{3})\approx$
	8	3.305×10^{4}	3.482×10^{4}	3.490×10^{4}	4.471×10^{4}	$\mathbf{2.847\times10^{4}}$	5.022×10^{4}	4.770×10^{4}	3.268×10^{4}
		(8.4×10^{3})	$(8.4\times10^{3})\approx$	$(7.6\times10^{3})\approx$	$(1.8\times10^{4})-$	$\mathbf{(9.6\times10^{3})\approx}$	$(1.0\times10^{4})-$	$(1.6\times10^{4})-$	$(7.4\times10^{3})\approx$
	10	3.545×10^{4}	3.599×10^{4}	3.980×10^{4}	4.835×10^{4}	$\mathbf{3.065\times10^{4}}$	3.246×10^{4}	5.216×10^{4}	3.455×10^{4}
		(4.5×10^{3})	$(6.3\times10^{3})\approx$	$(8.8\times10^{3})\approx$	$(8.0\times10^{3})-$	$\mathbf{(6.0\times10^{3})+}$	$(5.90\times10^{3})\approx$	$(6.9\times10^{3})-$	$(8.2\times10^{3})\approx$
LSMOP8	3	$\mathbf{3.940\times10^{-1}}$	4.540×10^{-1}	4.006×10^{-1}	4.445×10^{-1}	4.205×10^{-1}	1.071×10^{0}	4.654×10^{-1}	4.081×10^{-1}
		$\mathbf{(8.9\times10^{-2})}$	$(5.3\times10^{-2})-$	$(6.0\times10^{-2})\approx$	$(7.2\times10^{-2})-$	$(6.5\times10^{-2})\approx$	$(1.8\times10^{-1})-$	$(7.6\times10^{-2})-$	$(8.3\times10^{-2})\approx$
	5	$\mathbf{5.216\times10^{-1}}$	6.430×10^{-1}	6.342×10^{-1}	8.661×10^{-1}	6.569×10^{-1}	2.096×10^{0}	8.376×10^{-1}	6.559×10^{-1}
		$\mathbf{(1.3\times10^{-1})}$	$(7.1\times10^{-2})-$	$(1.1\times10^{-1})-$	$(1.3\times10^{-1})-$	$(1.2\times10^{-1})-$	$(2.8\times10^{-1})-$	$(1.3\times10^{-1})-$	$(1.1\times10^{-1})-$
	8	$\mathbf{2.282\times10^{0}}$	3.190×10^{0}	3.522×10^{0}	4.130×10^{0}	3.117×10^{0}	3.437×10^{0}	4.167×10^{0}	3.361×10^{0}
		$\mathbf{(6.5\times10^{-1})}$	$(4.8\times10^{-1})-$	$(5.1\times10^{-1})-$	$(6.7\times10^{-1})-$	$(5.0\times10^{-1})-$	$(4.4\times10^{-1})-$	$(3.1\times10^{-1})-$	$(4.3\times10^{-1})-$
	10	2.307×10^{0}	2.924×10^{0}	2.958×10^{0}	3.363×10^{0}	2.690×10^{0}	$\mathbf{2.299\times10^{0}}$	3.322×10^{0}	2.853×10^{0}
		(2.4×10^{-1})	$(3.4\times10^{-1})-$	$(3.6\times10^{-1})-$	$(2.9\times10^{-1})-$	$(4.0\times10^{-1})-$	$\mathbf{(2.1\times10^{-1})\approx}$	$(2.9\times10^{-1})-$	$(4.5\times10^{-1})-$
LSMOP9	3	4.151×10^{-1}	4.150×10^{-1}	4.220×10^{-1}	4.240×10^{-1}	$\mathbf{3.690\times10^{-1}}$	5.607×10^{-1}	4.334×10^{-1}	4.134×10^{-1}
		(8.1×10^{-2})	$(5.9\times10^{-2})\approx$	$(1.0\times10^{-1})\approx$	$(8.7\times10^{-2})\approx$	$\mathbf{(5.8\times10^{-2})+}$	$(1.3\times10^{-1})-$	$(9.4\times10^{-2})\approx$	$(6.4\times10^{-2})\approx$
	5	$\mathbf{2.658\times10^{-1}}$	2.945×10^{-1}	2.915×10^{-1}	3.451×10^{-1}	2.772×10^{-1}	4.072×10^{-1}	3.220×10^{-1}	2.750×10^{-1}
		$\mathbf{(3.9\times10^{-2})}$	$(3.7\times10^{-2})-$	$(5.0\times10^{-2})\approx$	$(5.9\times10^{-2})-$	$(4.2\times10^{-2})\approx$	$(7.1\times10^{-2})-$	$(5.3\times10^{2})-$	$(3.4\times10^{-2})\approx$
	8	$\mathbf{1.766\times10^{0}}$	2.421×10^{0}	3.296×10^{0}	2.265×10^{0}	3.525×10^{0}	4.172×10^{0}	2.267×10^{0}	3.801×10^{0}
		$\mathbf{(1.7\times10^{-1})}$	$(2.4\times10^{-1})-$	$(3.5\times10^{-1})-$	$(2.2\times10^{-1})-$	$(3.4\times10^{-1})-$	$(3.9\times10^{-1})-$	$(2.5\times10^{-1})-$	$(4.0\times10^{-1})-$
	10	$\mathbf{1.789\times10^{0}}$	2.766×10^{0}	4.112×10^{0}	2.824×10^{0}	4.783×10^{0}	4.820×10^{0}	2.740×10^{0}	4.940×10^{0}
		$\mathbf{(2.5\times10^{-1})}$	$(2.5\times10^{-1})-$	$(3.2\times10^{-1})-$	$(3.4\times10^{-1})-$	$(3.5\times10^{-1})-$	$(2.8\times10^{-1})-$	$(3.5\times10^{-1})-$	$(3.6\times10^{-1})-$
$+/-/\approx$			0/25/11	0/22/14	0/28/8	2/25/9	0/30/6	1/27/8	2/24/10

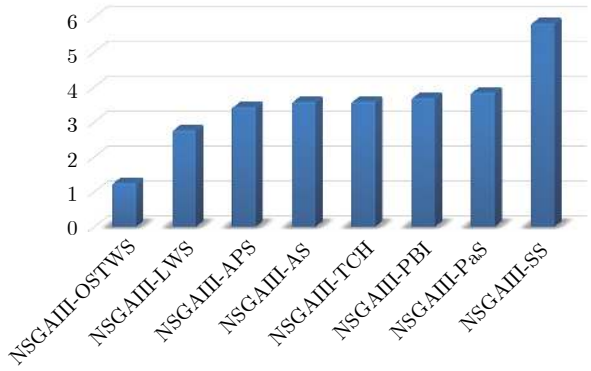


图 4 NSGAIII-OSTWS, NSGAIII-LWS, NSGAIII-TCH, NSGAIII-PBI, NSGAIII-AS, NSGAIII-SS, NSGAIII-APS 和 NSGAIII-PaS, 在所有测试问题实例中的平均 IGD$^+$性能得分排名. 得分越小, 整体性能越好

Fig. 4 Ranking in the average performance score over all test problem instances for the algorithms of NSGAIII-OSTWS, NSGAIII-LWS, NSGAIII-TCH, NSGAIII-PBI, NSGAIII-AS, NSGAIII-SS, NSGAIII-APS and NSGAIII-PaS. The smaller the score, the better the overall performance in terms of IGD$^+$

模决策变量问题处理上仍具有显著的优势, 能较快地将种群进化至 PF. 从表 5 可以看出, 本文所提出的算法 NSGAIII-OSTWS 在 LSMOP 的大部分测试问题上都取得了最佳的 GD 指标值, 这再次验证了 OSTWS 方法具有很强的搜索效率. 综合统计结果可以看出, NSGAIII-OSTWS 在 DTLZ、WFG 和 LSMOP 测试集上虽然没能在每个测试实例上都获得最优的 GD 结果, 但总体收敛性能优异.

为进一步测试本文所提出的 OSTWS 在多样性维持上的性能, 我们在前沿面为线性 (DTLZ1)、凹型 (DTLZ2)、退化 (DTLZ5) 和不连续 (DTLZ7) 的问题上, 将 NSGAIII-OSTWS 与其他 7 个算法进行对比. 表 6 展示了上述算法在最新多样性评价指标 CPF 上的平均测试结果, 可以看出 OSTWS 在整体上取得了最佳的多样性性能.

为直观地展示算法在平衡收敛性和多样性上的综合性能, 图 4 给出了各个算法在 IGD$^+$上的性能打分图, 分值越小表示该算法整体性能越好. 从图 4 可以看出, 算法 NSGAIII-OSTWS 获得了最小的 IGD$^+$打分值, 这表明与 NSGAIII 的变体相比, NSGAIII-OSTWS 具有很强的综合性能.

2) 算法整体性能验证与分析

为测试算法 NSGAIII-OSTWS 的综合性能, 将 NSGAIII-OSTWS 与 9 个先进的 MaOEAs 进行对比实验, 分别为 NSGA-III[25]、Two_arch2[27]、SRA[29]、SPEAR[26]、DDEANS[33]、HpaEA[28]、AR-MOEA[22]、MaOEA-IT[61] 和 PaRP/EA[62]. 表 7、表 8 和表 9 分别为上述算法在 DTLZ、WFG 和 LSMOP 测试问题集上所获得的 IGD$^+$统计结果.

从表 7, 表 8 和表 9 的数据可以看出, 与其它先进的 MaOEAs 相比, NSGAIII-OSTWS 在绝大部分测试问题上取得了最佳 IGD$^+$值, 这表明 NSGAIII-OSTWS 在平衡收敛性和多样性上非常具有竞争力. 首先, 由表 7, 表 8 和表 9 可以看出, NSGAIII-OSTWS 的性能要明显优于经典算法 NSGAIII, 说

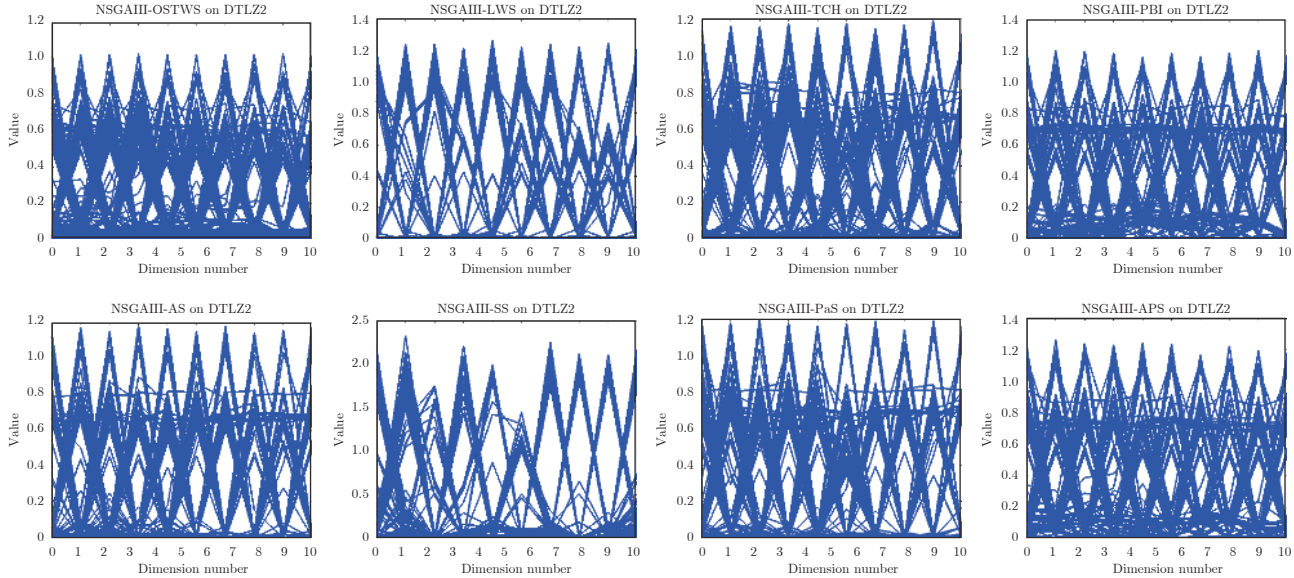


图 5 NSGAIII-OSTWS, NSGAIII-LWS, NSGAIII-TCH, NSGAIII-PBI, NSGAIII-AS, NSGAIII-SS, NSGAIII-PaS 和 NSGAIII-APS 在 10 维 DTLZ2 问题上所获得的解集

Fig. 5 Solution set of NSGAIII-OSTWS, NSGAIII-LWS, NSGAIII-TCH, NSGAIII-PBI, NSGAIII-AS, NSGAIII-SS, NSGAIII-PaS and NSGAIII-APS on DTLZ2 problem with 10-objectives

表 6　OSTWS, LWS, TCH, PBI, AS, SS, PaS 和 APS 方法在框架为 NSGAIII, 测试问题集为 DTLZ1, DTLZ2, DTLZ5 和 DTLZ7 上获得的 CPF 值统计结果 (均值和标准差). 每个实例算法中的最好结果以加粗突出显示

Table 6　The statistical results (mean and standard deviation) of the CPF values obtained by OSTWS, LWS, TCH, PBI, AS, SS, PaS and APS methods on the NSGAIII framework and DTLZ1, DTLZ2, DTLZ5 and DTLZ7 test problems. The best average value among the algorithms for each instance is highlighted in bold

Problem	m	NSGAIII-OSTWS	NSGAIII-LWS	NSGAIII-TCH	NSGAIII-PBI	NSGAIII-AS	NSGAIII-SS	NSGAIII-PaS	NSGAIII-APS
DTLZ1	3	**1.374×10^{-3}**	5.495×10^{-4}	8.242×10^{-4}	4.558×10^{-4}	1.099×10^{-3}	0.000×10^{0}	2.748×10^{-4}	4.021×10^{-4}
		(2.4×10^{-3})	$(1.7\times10^{-3})\approx$	$(2.7\times10^{-3})\approx$	$(1.4\times10^{-3})\approx$	$(2.3\times10^{-3})\approx$	$(0.0\times10^{0})-$	$(1.2\times10^{-3})\approx$	$(1.3\times10^{-3})\approx$
	5	1.436×10^{-4}	8.930×10^{-5}	2.924×10^{-4}	2.837×10^{-4}	2.954×10^{-4}	5.953×10^{-5}	**3.020×10^{-4}**	2.339×10^{-4}
		(3.5×10^{-4})	$(2.2\times10^{-4})\approx$	$(9.7\times10^{-4})\approx$	$(5.6\times10^{-4})\approx$	$(5.9\times10^{-4})\approx$	$(2.7\times10^{-4})\approx$	**$(5.0\times10^{-4})\approx$**	$(4.1\times10^{-4})\approx$
	8	7.099×10^{-5}	3.072×10^{-5}	1.083×10^{-4}	**3.749×10^{-4}**	5.327×10^{-5}	2.757×10^{-5}	1.655×10^{-4}	1.068×10^{-4}
		(1.1×10^{-4})	$(6.0\times10^{-5})\approx$	$(2.0\times10^{-4})\approx$	**$(7.7\times10^{-4})\approx$**	$(1.1\times10^{-4})\approx$	$(6.4\times10^{-5})\approx$	$(4.5\times10^{-4})\approx$	$(2.0\times10^{-4})\approx$
	10	**1.657×10^{-4}**	7.355×10^{-5}	6.571×10^{-6}	1.422×10^{-4}	4.603×10^{-5}	1.399×10^{-4}	7.511×10^{-5}	8.130×10^{-5}
		(4.1×10^{-4})	$(2.2\times10^{-4})\approx$	$(1.7\times10^{-5})-$	$(4.2\times10^{-4})\approx$	$(1.2\times10^{-4})\approx$	$(4.4\times10^{-4})\approx$	$(2.4\times10^{-4})\approx$	$(2.3\times10^{-4})\approx$
DTLZ2	3	5.698×10^{-1}	3.218×10^{-1}	5.792×10^{-1}	**6.891×10^{-1}**	5.427×10^{-1}	1.684×10^{-1}	5.632×10^{-1}	6.837×10^{-1}
		(4.3×10^{-2})	$(2.3\times10^{-2})-$	$(3.5\times10^{-2})\approx$	**$(1.1\times10^{-2})+$**	$(4.5\times10^{-2})-$	$(3.6\times10^{-2})-$	$(3.1\times10^{-2})\approx$	$(2.3\times10^{-2})+$
	5	5.993×10^{-1}	1.585×10^{-1}	5.521×10^{-1}	**7.114×10^{-1}**	5.416×10^{-1}	1.307×10^{-1}	5.433×10^{-1}	7.108×10^{-1}
		(2.2×10^{-2})	$(1.2\times10^{-2})-$	$(4.1\times10^{-2})-$	**$(1.4\times10^{-2})+$**	$(4.7\times10^{-2})-$	$(3.0\times10^{-2})-$	$(4.1\times10^{-2})-$	$(1.5\times10^{-2})+$
	8	3.780×10^{-1}	5.395×10^{-2}	2.871×10^{-1}	**4.085×10^{-1}**	2.922×10^{-1}	3.258×10^{-2}	2.947×10^{-1}	3.682×10^{-1}
		(2.8×10^{-2})	$(1.6\times10^{-2})-$	$(4.2\times10^{-2})-$	**$(2.8\times10^{-2})+$**	$(2.4\times10^{-2})-$	$(2.6\times10^{-2})-$	$(2.4\times10^{-2})-$	$(1.1\times10^{-1})\approx$
	10	**2.185×10^{-1}**	2.729×10^{-2}	1.752×10^{-1}	1.914×10^{-1}	1.912×10^{-1}	3.958×10^{-2}	1.855×10^{-1}	1.900×10^{-1}
		(3.7×10^{-3})	$(1.5\times10^{-2})-$	$(4.3\times10^{-2})-$	$(6.6\times10^{-2})\approx$	$(2.1\times10^{-2})-$	$(1.3\times10^{-2})-$	$(1.9\times10^{-2})-$	$(6.3\times10^{-2})\approx$
DTLZ5	3	6.043×10^{-1}	5.616×10^{-1}	5.755×10^{-1}	6.053×10^{-1}	5.639×10^{-1}	5.423×10^{-1}	5.925×10^{-1}	**6.092×10^{-1}**
		(4.4×10^{-2})	$(7.6\times10^{-2})-$	$(7.9\times10^{-2})\approx$	$(4.6\times10^{-2})\approx$	$(5.4\times10^{-2})-$	$(5.4\times10^{-2})-$	$(5.9\times10^{-2})\approx$	**$(5.3\times10^{-2})\approx$**
	5	**5.397×10^{-1}**	4.781×10^{-1}	3.670×10^{-1}	4.987×10^{-1}	2.654×10^{-1}	1.838×10^{-1}	2.452×10^{-1}	4.935×10^{-1}
		(7.5×10^{-2})	$(4.9\times10^{-2})-$	$(5.6\times10^{-2})-$	$(4.5\times10^{-2})-$	$(6.3\times10^{-2})-$	$(4.1\times10^{-2})-$	$(7.8\times10^{-2})-$	$(6.0\times10^{-2})-$
	8	**5.903×10^{-1}**	4.791×10^{-1}	5.093×10^{-1}	5.213×10^{-1}	4.770×10^{-1}	3.355×10^{-1}	3.963×10^{-1}	5.067×10^{-1}
		(1.2×10^{-1})	$(7.8\times10^{-2})-$	$(6.2\times10^{-2})-$	$(8.9\times10^{-2})\approx$	$(9.3\times10^{-2})-$	$(1.3\times10^{-1})-$	$(9.9\times10^{-2})-$	$(8.9\times10^{-2})-$
	10	**3.857×10^{-1}**	2.622×10^{-1}	3.066×10^{-1}	3.804×10^{-1}	2.635×10^{-1}	3.790×10^{-1}	2.391×10^{-1}	3.524×10^{-1}
		(5.1×10^{-2})	$(5.2\times10^{-2})-$	$(5.6\times10^{-2})-$	$(4.8\times10^{-2})\approx$	$(5.7\times10^{-2})-$	$(1.4\times10^{-1})\approx$	$(8.5\times10^{-2})-$	$(4.4\times10^{-2})-$
DTLZ7	3	**2.961×10^{-1}**	2.502×10^{-1}	2.853×10^{-1}	2.866×10^{-1}	2.835×10^{-1}	1.519×10^{-1}	2.676×10^{-1}	2.911×10^{-1}
		(4.3×10^{-2})	$(4.3\times10^{-2})-$	$(5.1\times10^{-2})\approx$	$(3.9\times10^{-2})\approx$	$(6.6\times10^{-2})\approx$	$(3.6\times10^{-2})-$	$(5.6\times10^{-2})\approx$	$(4.8\times10^{-2})\approx$
	5	2.716×10^{-1}	1.956×10^{-1}	2.760×10^{-1}	2.890×10^{-1}	2.622×10^{-1}	2.139×10^{-1}	2.530×10^{-1}	**2.974×10^{-1}**
		(3.4×10^{-2})	$(2.8\times10^{-2})-$	$(2.3\times10^{-2})\approx$	$(4.2\times10^{-2})\approx$	$(2.7\times10^{-2})\approx$	$(3.5\times10^{-2})-$	$(1.8\times10^{-2})\approx$	**$(3.1\times10^{-2})+$**
	8	**5.846×10^{-1}**	2.044×10^{-1}	3.897×10^{-1}	5.149×10^{-1}	3.996×10^{-1}	2.534×10^{-1}	3.618×10^{-1}	5.210×10^{-1}
		(1.0×10^{-1})	$(3.4\times10^{-2})-$	$(5.4\times10^{-2})-$	$(4.7\times10^{-2})-$	$(6.1\times10^{-2})-$	$(3.7\times10^{-2})-$	$(7.0\times10^{-2})-$	$(4.9\times10^{-2})-$
	10	1.3102×10^{-1}	2.657×10^{-1}	9.318×10^{-2}	1.994×10^{-1}	9.436×10^{-2}	**2.663×10^{-1}**	1.056×10^{-1}	2.015×10^{-1}
		(4.1×10^{-2})	$(3.2\times10^{-2})+$	$(2.0\times10^{-2})-$	$(1.6\times10^{-2})+$	$(1.5\times10^{-2})-$	**$(3.3\times10^{-2})+$**	$(2.4\times10^{-2})-$	$(2.0\times10^{-2})+$
$+/-/\approx$			1/11/4	0/9/7	4/2/10	0/10/6	1/11/4	0/8/8	4/4/8

表 7　NSGAIII-OSTWS, NSGAIII, Two_arch2, SRA, SPEAR, DDEANS, HpaEA, ARMOEA, MaOEA-IT 和 PaRP/EA 在 DTLZ1-7 上上获得的 IGD^{+}值的统计结果

Table 7　The statistical results of the IGD^{+} values obtained by NSGAIII-OSTWS, NSGAIII, Two_arch2, SRA, SPEAR, DDEANS, hpaEA, ARMOEA, MaOEA-IT and PaRP/EA on DTLZ1-7

NSGAIII-OSTWS	vs	NSGAIII	Two_Arch2	SRA	SPEAR	DDEANS	HpaEA	ARMOEA	MaOEA-IT	PaRP/EA
+		0/28	1/28	5/28	1/28	2/28	2/28	2/28	2/28	1/28
−		27/28	26/28	22/28	25/28	24/28	25/28	24/28	26/28	23/28
≈		1/28	1/28	1/28	2/28	2/28	1/28	2/28	0/28	4/28

明 OSTWS 在很大程度上提高了 NSGAIII 的整体性能, 这再次验证了 OSTWS 的有效性. 其次, NSGAIII-OSTWS 的整体性能要强于其它最新算法 Two_arch2、SRA、SPEAR、DDEANS、HpaEA、ARMOEA、MaOEA-IT 和 PaRP/EA, 尤其是在规则问题上, 如 DTLZ1-4 和 WFG4-9. 原因是 NSGAIII-OSTWS 在规则问题上易于预估出 PF 的形状, 从而有利于加速种群的收敛. 值得注意的是, DDEANS 也获得了较好的性能, 原因是 DDEANS 采用了参考向量调整的方法, 从而能在不规则测试问题上, 如 DTLZ7 和 WFG1-2, 较好地维持种群的多样性.

表 8 NSGAIII-OSTWS, NSGAIII, Two_arch2, SRA, SPEAR, DDEANS, HpaEA, ARMOEA, MaOEA-IT 和 PaRP/EA 在 WFG1-9 上上获得的 IGD$^+$值的统计结果

Table 8 The statistical results of the IGD$^+$ values obtained by NSGAIII-OSTWS, NSGAIII, Two_arch2, SRA, SPEAR, DDEANS, HpaEA, ARMOEA, MaOEA-IT and PaRP/EA on WFG1-9

NSGAIII-OSTWS	vs	NSGAIII	Two_Arch2	SRA	SPEAR	DDEANS	HpaEA	ARMOEA	MaOEA-IT	PaRP/EA
+		1/36	0/36	0/36	0/36	5/36	0/36	0/36	0/36	0/36
–		35/36	26/36	35/36	35/36	30/36	36/36	36/36	36/36	34/36
≈		0/36	0/36	1/36	1/36	1/36	0/36	0/36	0/36	2/36

表 9 NSGAIII-OSTWS, NSGAIII, Two_arch2, SRA, SPEAR, DDEANS, HpaEA, ARMOEA, MaOEA-IT 和 PaRP/EA 在 LSMOP1-9 上获得的 IGD$^+$值的统计结果

Table 9 The statistical results of the IGD$^+$ values obtained by NSGAIII-OSTWS, NSGAIII, Two_arch2, SRA, SPEAR, DDEANS, HpaEA, ARMOEA, MaOEA-IT and PaRP/EA on LSMOP1-9

NSGAIII-OSTWS	vs	NSGAIII	Two_Arch2	SRA	SPEAR	DDEANS	HpaEA	ARMOEA	MaOEA-IT	PaRP/EA
+		10/36	13/36	12/36	10/36	11/36	10/36	16/36	7/36	9/36
-		21/36	21/36	20/36	23/36	22/36	23/36	17/36	24/36	23/36
≈		5/36	2/36	4/36	3/36	3/36	3/36	3/36	5/36	4/36

为直观展示各个算法在平衡种群收敛性和多样性上的能力, 图 6 给出了所有算法在 10 维 DTLZ4 问题上的最终种群分布图. 从图 6 可以看出, 算法 NSGAIII-OSTWS 获得的种群具有良好的收敛性和多样性, 而对比算法 NSGA-III、Two_arch2、SRA、SPEAR、DDEANS、HpaEA、ARMOEA 和 MaOEA-IT 获得的种群都没有收敛到 PF, 这反映了 NSGAIII-OSTWS 的优越性. 此外, 为进一步观察各个算法的收敛性能, 图 7 给出了各个算法在 DTLZ、WFG 和 LSMOP 测试问题上的平均 GD 表现分, 分值越小表示该算法的收敛性性能越好. 从图 7 可以看出, 算法 NSGAIII-OSTWS 在绝大部分测试问题上都获得了最低的 GD 表现分, 表明与其它类型的算法相比, NSGAIII-OSTWS 具有很强的收敛能力, 这归功于 NSGAIII-OSTWS 算法采用的 OSTWS 方法具有很高的搜索效率, 从而使得种群能快速的收敛到 PF. 但对于退化和不连续的不规则测试问题 DTLZ5-7, 由于较难准确预估出真实 PF 的形状, 导致 NSGAIII-OSTWS 在处理 DTLZ5-7 问题时收敛性会有所下降.

3) 参数 C 的敏感性分析

NSGAIII-OSTWS 算法的核心是将各种问题的 PF 转换为凸型曲面, 其中参数 C 用于控制预设凸曲面的曲率, 因此对算法的性能有着一定的影响.

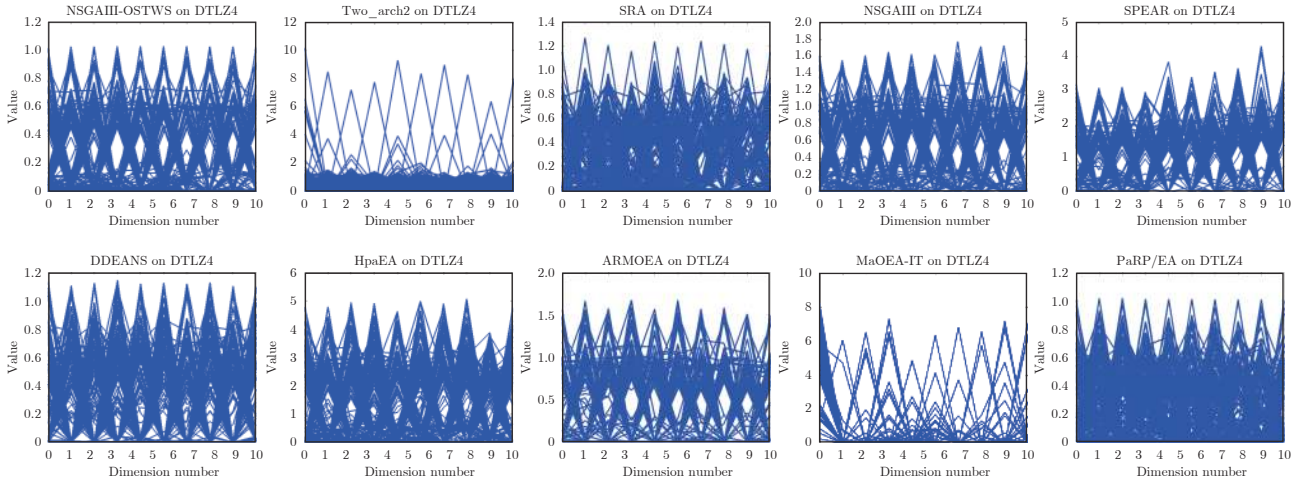


图 6 SRA, SPEAR, DDEANS, HpaEA, ARMOEA, MaOEA-IT 和 PaRP/EA 在 10 维 DTLZ4 问题上所获得的解集

Fig. 6 Solution set of NSGAIII-OSTWS, NSGAIII, Two_arch2, SRA, SPEAR, DDEANS HpaEA, ARMOEA, MaOEA-IT and PaRP/EA on DTLZ4 problem with 10-objectives

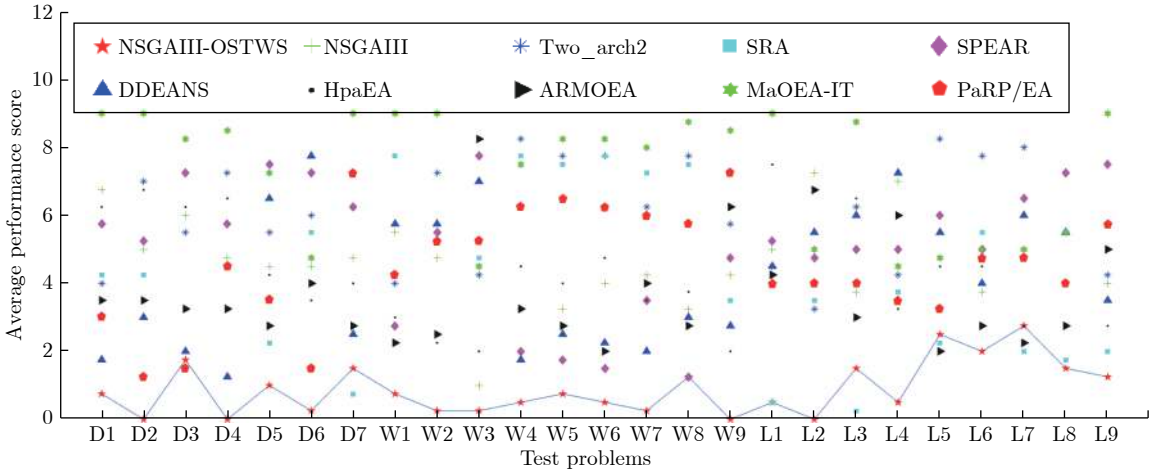


图 7　NSGAIII-OSTWS, NSGAIII, Two_arch2, SRA, SPEAR, DDEANS, HpaEA, ARMOEA, MaOEA-IT 和 PaRP/EA 在所有测试问题, 即 DTLZ(Dx), WFG(Wx) 和 LSMOP(Lx) 上的平均 GD 表现分, 分值越小, 算法的整体性能越好. 通过实线连接 NSGAIII-OSTWS 的得分, 以便易于评估分数

Fig. 7　Average performance score of NSGAIII-OSTWS, NSGAIII, Two_arch2, SRA, SPEAR, DDEANS, HpaEA, ARMOEA, MaOEA-IT and PaRP/EA on all test problems, namely DTLZ(Dx), WFG(Wx)and LSMOP(Lx). The smaller the score, the better the overall performance in terms of GD. The values of NSGAIII-OSTWS are connected by a solid line to easier assess the score

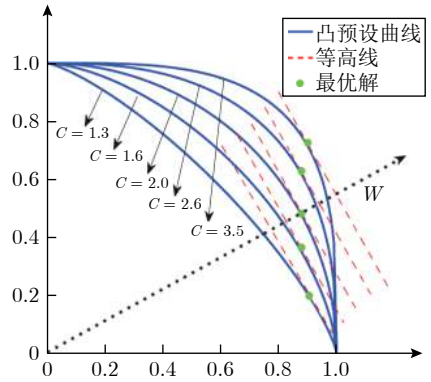


图 8　不同预设凸曲线在参考向量 $\boldsymbol{w}$ 上获得的最优解示意图

Fig. 8　The optimal solution obtained by different preset convex curves on the reference vector $\boldsymbol{w}$

由于预设曲面为凸形, 其曲率值 $C > 1$. 图 8 为不同预设凸曲线在参考向量 $\boldsymbol{w}$ 上获得的最优解示意图. 从中可以看出, 当 $1 < C < 2$ 时, 由参考向量 $\boldsymbol{w}$ 获得的最优解逐渐靠近 $\boldsymbol{w}$; 当 $C = 2$ 时, $\boldsymbol{w}$ 获得的最优解正好位于 $\boldsymbol{w}$ 上; 当 $C > 2$ 时, $\boldsymbol{w}$ 获得的最优解逐渐远离 $\boldsymbol{w}$. 由于参考向量主要用于对种群多样性的管理, 最优解越靠近参考向量越有利于多样性的维护, 因此 C 值取 2 相比其它取值更能维持种群的多样性. 为验证这一结论, 本小节在凹凸混合问题 WFG1、凹问题 WFG4、线性问题 DTLZ1 和不连续问题 DTLZ7 上测试了 C 为{1.2, 2, 3, 4, 5, 6, 7, 8, 9, 10}的 IGD$^+$性能. 图 9 为实验结果, 可以看出, 随着 C 值的增加, NSGAIII-OSTWS 的整体性能先逐渐提升后逐渐恶化, 当 $C = 2$ 时, 算法具有最佳的性能, 这验证了当 C 值取 2 时算法具备良好性能这一结论.

4　结论

为充分发挥权重求和搜索效率高的优势, 同时又能处理好 PF 形状为非凸的问题, 本文提出了一种基于目标空间转换权重求和的超多目标进化算法, 简称 NSGAIII-OSTWS, 其中目标空间转换权重求和方法 (OSTWS) 将各种问题的 PF 转换为凸型曲面, 再对其进行优化求解. 为验证 NSGAIII-OSTWS 的有效性, NSGAIII-OSTWS 与 7 个 NSGAIII 的变体以及 9 个最新的 MaOEAs 在 WFG、DTLZ 和 LSMOP 测试问题集上进行了实验对比. 实验结果表明, 相比其他算法, NSGAIII-OSTWS 具备良好的竞争性能.

虽然所提算法 NSGAIII-OSTWS 在处理 MaOPs 上取得了优越的性能, 但仍有一些开放性的工作值得进一步的探索. 例如, 不规则问题的 PF 形状难以精确预估, 这势必导致个体映射的不准确进而影响算法的性能. 因此, 未来需要进一步研究如何更加精确地映射个体. 此外, 将本文所提出的算法应用于现实生产和生活中各类实际问题的求解具有重要的价值, 这也是未来需要进一步开展的工作. 本文所提算法的源代码已在 https://github.com/CIA-SZU/LTT 上公开.

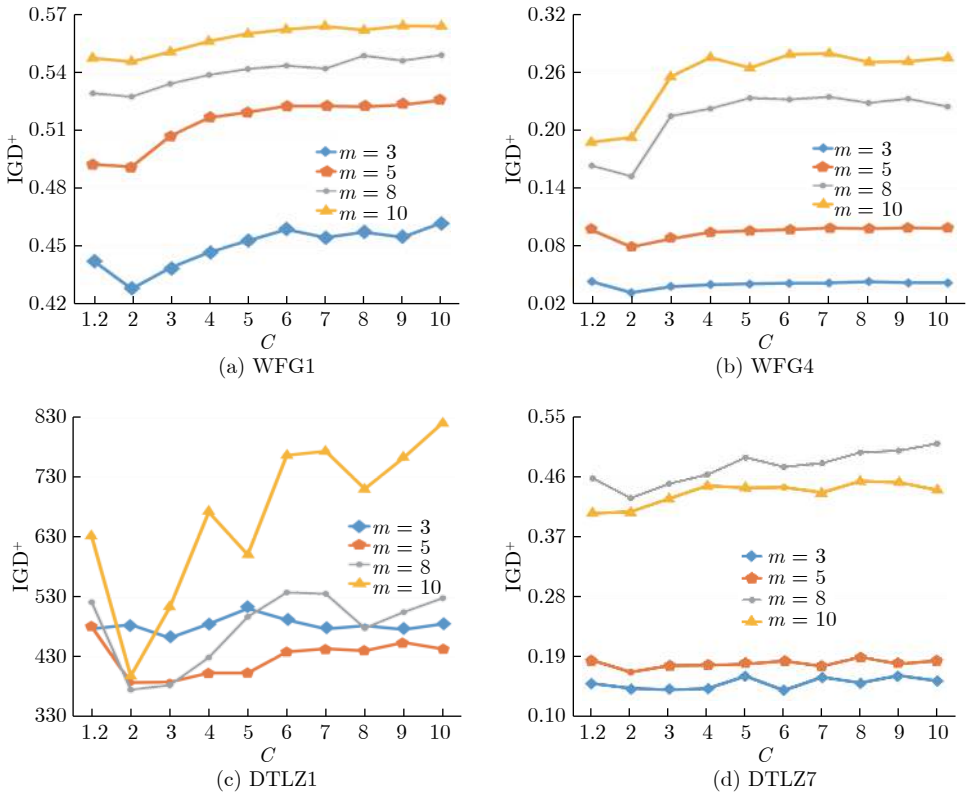


图 9 不同 C 值在 WFG1, WFG4, DTLZ1 和 DTLZ7 问题的 3, 5, 8 和 10 目标维度上的 IGD$^+$均值

Fig. 9 The median IGD$^+$ values of different C values on WFG1, WFG4, DTLZ1 and DTLZ7 problems with 3-, 5-, 8-, and 10-objectives

References

1 Kong Wei-Jian, Chai Tian-You, Ding Jin-Liang, Wu Zhi-Wei. A real-time multiobjective electric energy allocation optimization approach for the smelting process of magnesia. *Acta Automatica Sinica*, 2014, **40**(1): 51−61
(孔维健, 柴天佑, 丁进良, 吴志伟. 镁砂熔炼过程全厂电能分配实时多目标优化方法研究. 自动化学报, 2014, **40**(1): 51−61)

2 Qiao Jun-Fei, Han Gai-Tang, Zhou Hong-Biao. Knowledge-based intelligent optimal control for wastewater biochemical treatment process. *Acta Automatica Sinica*, 2017, **43**(6): 1038−1046
(乔俊飞, 韩改堂, 周红标. 基于知识的污水生化处理过程智能优化方法. 自动化学报, 2017, **43**(6): 1038−1046)

3 Lin Chuang, Chen Ying, Huang Ji-Wei, Xiang Xu-Dong. A survey on models and solutions of multi-objective optimization for QoS in services computing. *Chinese Journal of Computers*, 2015, **38**(10): 1907−1923
(林闯, 陈莹, 黄霁崴, 向旭东. 服务计算中服务质量的多目标优化模型与求解研究. 计算机学报, 2015, **38**(10): 1907−1923)

4 Wang J H, Zhou Y, Wang Y, Zhang J, Chen C L P, Zheng Z B. Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: Formulation, instances, and algorithms. *IEEE Transactions on Cybernetics*, 2016, **46**(3): 582−594

5 Sarro F, Ferrucci F, Harman M, Manna A, Ren J. Adaptive multi-objective evolutionary algorithms for overtime planning in software projects. *IEEE Transactions on Software Engineering*, 2017, **43**(10): 898−917

6 Cao B, Dong W N, Lv Z H, Gu Y, Singh S, Kumar P. Hybrid microgrid many-objective sizing optimization with fuzzy decision. *IEEE Transactions on Fuzzy Systems*, 2020, **28**(11): 2702−2710

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

8 Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Swiss Federal Institute of Technology, Switzerland, TIK-Report 103, 2001. DOI: 10.3929/ETHZ-A-004284029

9 Feng Wen-Qing, Gong Dun-Wei. Multi-objective evolutionary optimization with objective space partition based on online perception of Pareto front. *Acta Automatica Sinica*, 2020, **46**(8): 1628−1643
(封文清, 巩敦卫. 基于在线感知Pareto前沿划分目标空间的多目标进化优化. 自动化学报, 2020, **46**(8): 1628−1643)

10 Purshouse R C, Fleming P J. On the evolutionary optimization of many conflicting objectives. *IEEE Transactions on Evolutionary Computation*, 2007, **11**(6): 770−784

11 Wang Li-Ping, Zhang Ming-Lei, Qiu Fei-Yue, Jiang Bo. Many-objective optimization algorithm with preference based on the angle penalty distance elite selection strategy. *Chinese Journal of Computers*, 2018, **41**(1): 236−253
(王丽萍, 章鸣雷, 邱飞岳, 江波. 基于角度惩罚距离精英选择策略的偏好高维目标优化算法. 计算机学报, 2018, **41**(1): 236−253)

12 Xie Cheng-Wang, Yu Wei-Wei, Bi Ying-Zhou, Wang Shen-Wen, Hu Yu-Rong. Many-objective evolutionary algorithm based on decomposition and coevolution. *Journal of Software*, 2020, **31**(2): 356−373
(谢承旺, 余伟伟, 闭应洲, 汪慎文, 胡玉荣. 一种基于分解和协同的高维多目标进化算法. 软件学报, 2020, **31**(2): 356−373)

13 He Z N, Yen G G. Many-objective evolutionary algorithms based on coordinated selection strategy. *IEEE Transactions on Evolutionary Computation*, 2017, **21**(2): 220−233

14 Chen L, Liu H L, Tan K C, Cheung Y M, Wang Y P. Evolutionary many-objective algorithm using decomposition-based dominance relationship. *IEEE Transactions on Cybernetics*,

2019, **49**(12): 4129−4139

15 Pan L Q, He C, Tian Y, Wang H D, Zhang X Y, Jin Y C. A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2019, **23**(1): 74−88

16 Laumanns M, Thiele L, Deb K, Zitzler E. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 2002, **10**(3): 263−282

17 Zou X F, Chen Y, Liu M Z, Kang L S. A new evolutionary algorithm for solving many-objective optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 2008, **38**(5): 1402−1412

18 He Z N, Yen G G, Zhang J. Fuzzy-based Pareto optimality for many-objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2014, **18**(2): 269−285

19 Yu Wei-Wei, Xie Cheng-Wang, Bi Ying-Zhou, Xia Xue-Wen, Li Xiong, Ren Ke-Yan, et al. Many-objective particle swarm optimization based on adaptive fuzzy dominance. *Acta Automatica Sinica*, 2018, **44**(12): 2278−2289
(余伟伟, 谢承旺, 闭应洲, 夏学文, 李雄, 任柯燕, 等. 一种基于自适应模糊支配的高维多目标粒子群算法. 自动化学报, 2018, **44**(12): 2278−2289)

20 Zitzler E, Künzli S. Indicator-based selection in multiobjective search. In: Proceedings of the 8th International Conference on Parallel Problem Solving from Nature. Birmingham, UK: Springer, 2004. 832−842

21 Bader J, Zitzler E. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 2011, **19**(1): 45−76

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

23 Chen Guo-Yu, Li Jun-Hua, Li Ming, Chen Hao. An R2 indicator and reference vector based many-objective optimization evolutionary algorithm. *Acta Automatica Sinica*, 2021, **47**(11): 2675−2690
(陈国玉, 李军华, 黎明, 陈昊. 基于R2指标和参考向量的高维多目标进化算法. 自动化学报, 2021, **47**(11): 2675−2690)

24 Zhang Q F, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 2007, **11**(6): 712−731

25 Deb K, Jain H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 2014, **18**(4): 577−601

26 Jiang S Y, Yang S X. A strength Pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2017, **21**(3): 329−346

27 Wang H D, Jiao L C, Yao X. Two_arch2: An improved two-archive algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2015, **19**(4): 524−541

28 Chen H K, Tian Y, Pedrycz W, Wu G H, Wang R, Wang L. Hyperplane assisted evolutionary algorithm for many-objective optimization problems. *IEEE Transactions on Cybernetics*, 2020, **50**(7): 3367−3380

29 Li B D, Tang K, Li J L, Yao X. Stochastic ranking algorithm for many-objective optimization based on multiple indicators. *IEEE Transactions on Evolutionary Computation*, 2016, **20**(6): 924−5938

30 Hua Y C, Jin Y C, Hao K R. A clustering-based adaptive evolutionary algorithm for multiobjective optimization with irregular Pareto fronts. *IEEE Transactions on Cybernetics*, 2019, **49**(7): 2758−2770

31 Qi Y T, Ma X L, Liu F, Jiao L C, Sun J Y, Wu J S. MOEA/D with adaptive weight adjustment. *Evolutionary Computation*, 2014, **22**(2): 231−264

32 Asafuddoula M, Singh H K, Ray T. An enhanced decomposition-based evolutionary algorithm with adaptive reference vectors. *IEEE Transactions on Cybernetics*, 2018, **48**(8): 2321−2334

33 He X Y, Zhou Y R, Chen Z F, Zhang Q F. Evolutionary many-objective optimization based on dynamical decomposition. *IEEE Transactions on Evolutionary Computation*, 2019, **23**(3): 361−375

34 Ishibuchi H, Setoguchi Y, Masuda H, Nojima Y. Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 2017, **21**(2): 169−190

35 Hansen M P. Use of substitute scalarizing functions to guide a local search based heuristic: The case of moTSP. *Journal of Heuristics*, 2000, **6**(3): 419−431

36 Wang R, Zhang Q F, Zhang T. Decomposition-based algorithms using Pareto adaptive scalarizing methods. *IEEE Transactions on Evolutionary Computation*, 2016, **20**(6): 821−837

37 Ishibuchi H, Sakane Y, Tsukamoto N, Nojima Y. Adaptation of scalarizing functions in MOEA/D: An adaptive scalarizing function-based multiobjective evolutionary algorithm. In: Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization. Nantes, France: Springer, 2009. 438−452

38 Ishibuchi H, Sakane Y, Tsukamoto N, Nojima Y. Simultaneous use of different scalarizing functions in MOEA/D. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. Portland, USA: ACM, 2010, 519−526

39 Wang R, Zhou Z B, Ishibuchi H, Liao T J, Zhang T. Localized weighted sum method for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2018, **22**(1): 3−18

40 Li K, Zhang Q F, Kwong S, Li M Q, Wang R. Stable matching-based selection in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 2014, **18**(6): 909−923

41 Jiang S Y, Yang S X. An improved multiobjective optimization evolutionary algorithm based on decomposition for complex Pareto fronts. *IEEE Transactions on Cybernetics*, 2016, **46**(2): 421−437

42 Wang L P, Zhang Q F, Zhou A M, Gong M G, Jiao L C. Constrained subproblems in a decomposition-based multiobjective evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, 2016, **20**(3): 475−480

43 Ishibuchi H, DoiK, Nojima Y. Use of piecewise linear and nonlinear scalarizing functions in MOEA/D. In: Proceedings of the 14th International Conference on Parallel Problem Solving from Nature. Edinburgh, UK: Springer, 2016. 503−513

44 Sato H. Analysis of inverted PBI and comparison with other scalarizing functions in decomposition based MOEAs. *Journal of Heuristics*, 2015, **21**(6): 819−849

45 Zitzler E, Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 1999, **3**(4): 257−271

46 Liang Z P, Hu K F, Ma X L, Zhu Z X. A many-objective evolutionary algorithm based on a two-round selection strategy. *IEEE Transactions on Cybernetics*, 2021, **51**(3): 1417−1429

47 Yang S X, Li M Q, Liu X H, Zheng J H. A grid-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2013, **17**(5): 721−736

48 Yuan Y, Xu H, Wang B, Yao X. A new dominance relation-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2016, **20**(1): 16−37

49 Deb K, Agrawal R B. Simulated binary crossover for continuous search space. *Complex Systems*, 1995, **9**(2): 115−148

50 Deb K, Goyal M. A combined genetic adaptive search (GeneAS) for Engineering design. *Computer Science and Informatics*, 1996, **26**(4): 30−45

51 Huband S, Hingston P, Barone L, While L. A review of multiob-

jective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 2006, **10**(5): 477−506

52 Deb K, Thiele L, Laumanns M, Zitzler E. Scalable test problems for evolutionary multiobjective optimization. *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*. London: Springer, 2005. 105−145

53 Cheng R, Jin Y C, Olhofer M, Sendhoff B. Test problems for large-scale multiobjective and many-objective optimization. *IEEE Transactions on Cybernetics*, 2017, **47**(12): 4108−4121

54 Bosman P A N, Thierens D. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2003, **7**(2): 174−188

55 Tian Y, Cheng R, Zhang X Y, Li M Q, Jin Y C. Diversity assessment of multi-objective evolutionary algorithms: Performance metric and benchmark problems [Research Frontier]. *IEEE Computational Intelligence Magazine*, 2019, **14**(3): 61−74

56 Ishibuchi H, Masuda H, Nojima Y. A study on performance evaluation ability of a modified inverted generational distance indicator. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. Madrid, Spain: ACM, 2015. 695−702

57 Liang Z P, Luo T T, Hu K F, Ma X L, Zhu Z X. An indicator-based many-objective evolutionary algorithm with boundary protection. *IEEE Transactions on Cybernetics*, 2021, **51**(9): 4553−4566

58 Li K, Deb K, Zhang Q F, Kwong S. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, 2015, **19**(5): 694−716

59 Wilcoxon F. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1945, **1**(6): 80−83

60 Yang S X, Jiang S Y, Jiang Y. Improving the multiobjective evolutionary algorithm based on decomposition with new penalty schemes. *Soft Computing*, 2017, **21**(16): 4677−4691

61 Yu Y N, Xue B, Zhang M J, Yen G G. A new two-stage evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2019, **23**(5): 748−761

62 Xiang Y, Zhou Y R, Yang X W, Huang H. A many-objective evolutionary algorithm with Pareto-adaptive reference points. *IEEE Transactions on Evolutionary Computation*, 2020, **24**(1): 99−113

梁正平 深圳大学计算机与软件学院副教授. 2006 年获武汉大学博士学位. 主要研究方向为计算智能, 大数据分析与应用.
E-mail: liangzp@szu.edu.cn
(**LIANG Zheng-Ping** Associate professor at the School of Computer Science and Software Engineering, Shenzhen University. He received his Ph. D. degree from WuHan University in 2006. His research interest covers computational intelligence, big data analysis and application.)

骆婷婷 华为技术有限公司工程师. 2020 年获深圳大学硕士学位. 主要研究方向为计算智能, 自然语言处理与应用.
E-mail: luotingting2017@email.szu.edu.cn
(**LUO Ting-Ting** Engineer at Huawei Technology Co., Ltd. She received her master degree from Shenzhen University in 2020. Her research interest covers computational intelligence, natural language processing and applications.)

王志强 深圳大学计算机与软件学院教授. 主要研究方向为计算智能, 大数据分析与应用和多媒体技术与应用. E-mail: wangzq@szu.edu.cn
(**WANG Zhi-Qiang** Professor at the School of Computer Science and Software Engineering, Shenzhen University. His research interest covers computational intelligence, big data analysis and applications, and multimedia technology and applications.)

朱泽轩 深圳大学计算机与软件学院教授. 2008 年获新加坡南洋理工大学博士学位. 主要研究方向为计算智能, 机器学习与生物信息学. 本文通信作者.
E-mail: zhuzx@szu.edu.cn
(**ZHU Ze-Xuan** Professor at the School of Computer Science and Software Engineering, Shenzhen University. He received his Ph. D. degree from Nanyang Technological University in 2008. His research interest covers computational intelligence, machine learning and bioinformatics. Corresponding author of this paper.)

胡凯峰 深圳大学信息中心工程师. 2019 年获深圳大学硕士学位. 主要研究方向为计算智能及其应用.
E-mail: kaifeng@szu.edu.cn
(**HU Kai-Feng** Engineer at Information Center, Shenzhen University. He received his master degree from Shenzhen University in 2019. His research interest covers computational intelligence and applications.)